European Commission

eIDAS-Node Installation and Configuration Guide

Version 2.4

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
1.4	06/10/2017	• Restructuring of reference documentation • Modifications to remove support for JBoss6. • Support WebLogic 12.2 family of servers. • Amend filename conventions to change '\' to '/'.	DIGIT
2.0	11/04/2018	• Changes in supported application servers; • Configuration and stability improvements; • Architectural changes (separation of Specific Connector and Specific Proxy Service). (for details see the Version 2.0 Release Notes and the *eIDAS-Node Migration Guide*)	DIGIT
1	05/07/2018	Reuse of document policy updated and version changed to match the corresponding Release.	DIGIT
2.2	14/09/2018	Document updated to reflect current installation and configuration.	DIGIT
2.3	20/06/2019	Document updated to reflect current installation and configuration.	DIGIT

2.4	06/12/2019	Document updated to reflect current installation and configuration.	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 information of a technical nature and does not supplement or amend the terms and conditions of any procurement procedure; therefore, no compensation claim can be based on the contents of this document.

Table of contents

List of figures

List of tables

List of abbreviations

The following abbreviations are used within this document.

Abbreviation	**Meaning**
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	Middleware. Architecture of the integration of eIDs in services, with a direct communication between SP and the citizen's PC without any central server. The term also refers to the piece of software of this architecture that executes on the citizen's PC.
MS	Member State
SAML	Security Assertion Markup Language
SP	Service Provider

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
Basic Setup	The basic configuration and Demo tools provided in a package to setup and run an eIDAS-Node strictly for demo purposes only.
Demo tools	Demo tools comprise the Demo SP, Demo IDP, Specific Connector and Specific Proxy Service included in the integration package. These components are not production ready and should not be deployed or used in production environments.
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.

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

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 – *Product overview* describes the binaries and source code to be installed plus the configuration files.
- Chapter 3 – *Preparing the installation* describes the prerequisites for a successful installation, including the correct Java version, supported application servers, environmental variables to be set, keystores etc.
- Chapter 4 – *Configuring the software* describes all configuration settings.
- Chapter 5 – *Building and deploying the software* describes the steps to build and then to deploy the software on the supported servers. There are two main types of eIDAS-Node: Connector and Proxy Service.
- Chapter 6 — *Verifying the installation* shows the final structure of your application server relevant directories, so that you can confirm that you have made the proper configurations.
- Chapter 7 – *Advanced configuration for production environments* provides detailed descriptions of the configurations to enable you to change specific aspects as required.
- Appendix A – *eIDAS Levels of Assurance* provides information on the three Levels of Assurance described in the Implementing Regulation.
- Appendix B – *User consent* provides a brief overview of the meaning of 'user consent' in the context of privacy legislation.
- Appendix C – *Ignite proposed configuration* provides specific information related to configuration of a cluster environment using Ignite.
- Appendix D – *Hazelcast proposed configuration* describes aspects of Hazelcast propose configuration.
- Appendix E — *Installation Frequently Asked Questions* provides answers to questions that may arise during your installation.

1.2. Purpose

The purpose of this document is to give a comprehensive view of eID and its components (in terms of binaries, source code and configuration files).

1.3. 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;
- show how to ensure eIDAS regulation compliance and provide a check list of recommendations;
- describe the technologies and configurations used for testing the eIDAS-Node in cluster mode.

1.4. Other technical reference documentation

We recommend that you also familiarise yourself with the following eID technical reference documents which are available on **CEF Digital Home > eID > All eID services > eIDAS Node integration package > View latest version**:

- *eIDAS-Node Installation, Configuration and Integration Quick Start Guide* describes how to quickly install a demo Service Provider, eIDAS-Node Connector, eIDAS-Node Proxy Service and demo IdP from the distributions in the release package. The distributions provide preconfigured eIDAS-Node modules for running on each of the supported application servers.
- *eIDAS-Node National IdP and SP Integration Guide* provides guidance by recommending one way in which eID can be integrated into your national eID infrastructure.
- *eIDAS-Node Demo Tools Installation and Configuration Guide* describes the installation and configuration settings for Demo Tools (SP and IdP) supplied with the package for basic testing.
- *eIDAS-Node and SAML* 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.
- *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. It describes the files that are generated, the file format, the components that are monitored and the events that are recorded.
- *eIDAS-Node Security Considerations* describes the security considerations that should be taken into account when implementing and operating your eIDAS-Node scheme.
- *eIDAS-Node Error Codes* contains tables showing the error codes that could be generated by components along with a description of the error, specific behaviour and, where relevant, possible operator actions to remedy the error.

1.5. eIDAS Technical specifications and software provided

This software package is provided as a reference implementation in accordance with the *eIDAS Technical Specifications* v1.1 available at https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eIDAS+Profile.

1.5.1. Further information

For further information on the practical implementation of the features listed above, please refer to section 7.5 — *eIDAS-Node compliance* which describes the production mode for ensuring eIDAS regulation compliance.

Disclaimer: The users of the eIDAS-Node sample implementation remain fully responsible for its integration with back-end systems (Service Providers and Identity Providers), testing, deployment and operation. The support and maintenance of the sample implementation, as well as any other auxiliary services, are provided by the European Commission according to the terms defined in the European Union Public License (EUPL) at https://joinup.ec.europa.eu/sites/default/files/custom-page/attachment/eupl_v1.2_en.pdf

2. Product overview

2.1. Package

The main product deliverable is `EidasNode.war` which is a web application that can be deployed to most Java web containers on the market. Both the eIDAS-Node Connector and the eIDAS-Node Proxy Service are implemented in this package. The actual functionality is activated by configuration.

2.2. Modules

The software is composed of several modules. This section describes the binaries and source code to be installed plus the configuration files.

Table 1: 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 utility classes used for implementing as basis for the EIDAS-Commons and MS Specific Connector and MS Specific Proxy Service modules.
Commons	EIDAS-Commons	Common Applications components and utility classes for implementing functionality of authentication service.
Encryption	EIDAS-Encryption	Encryption and signature dedicated module. Contains libraries, including OpenSAML, provided for the build as Maven local repository.
Metadata	EIDAS-Metadata	Implementation of metadata related functionalities such as generation and fetching used in both EIDAS-SAMLEngine and eIDAS-Node.
ConfigModule	EIDAS-ConfigModule	Configuration management module dedicated to facilitate eIDAS-Node configuration.
SAMLEngine	EIDAS-SAMLEngine	Implementation of EIDAS SAML ProtocolEngine used in the eIDAS-Node.
JCache-Dev	EIDAS-JCache-Dev	Common code for Guava non-distributed JCache implementations
JCache-Dev-Node	EIDAS-JCache-Dev-Node	Adapts the implementation of Guava non-distributed maps to JCache used in eIDAS-Node caches.

Module Name	Folder	Description
JCache-Dev-Specific-Communication	EIDAS-JCache-Dev-Specific-Communication	Adapts the implementation of Guava non-distributed maps to JCache used in eIDAS-Node MS specific communication caches.
JCache-Ignite	EIDAS-JCache-Ignite	Common code for of Ignite JCache implementations
JCache-Ignite-Node	EIDAS-JCache-Ignite-Node	Implementation of Ignite JCache for the eIDAS-Node caches
JCache-Ignite-Specific-Communication	EIDAS-JCache-Ignite-Specific-Communication	Implementation of Ignite JCache for the eIDAS-Node MS specific communication caches
JCache-Hazelcast	EIDAS-JCache-Hazelcast	Common code for Hazelcast maps Jcache adaptation implementations
JCache-Hazelcast-Node	EIDAS-JCache-Hazelcast-Node	Implementation of Hazelcast maps Jcache adaptation for the eIDAS-Node caches
JCache-Hazelcast-Specific-Communication	EIDAS-JCache-Hazelcast-Specific-Communication	Implementation of Hazelcast maps Jcache adaptation for the eIDAS-Node MS specific communication caches
Specific Communication Definition	EIDAS-SpecificCommunicationDefinition	The exchange definition (interfaces) and implementation used to formalise the exchange definition between the node and the Specific module.
Updater	EIDAS-Updater	Module used to change configuration of a running eIDAS-Node in testing environment. (To enable, web.xml must be updated.) Not to be used in production.
EidasNode	EIDAS-NODE	eIDAS-Node module (Proxy Service, Connector).
Basic Setup configuration	EIDAS-Config	Sample configuration as in 6.7

The figure below shows the dependencies between the installed modules.

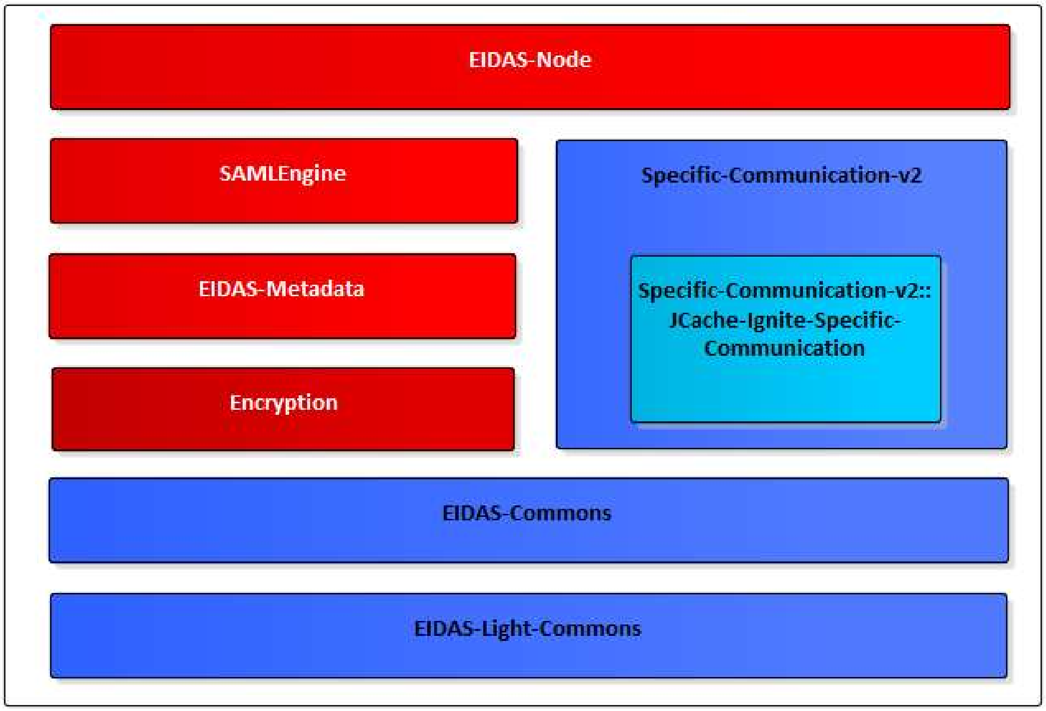

EIDAS-Node
SAMLEngine
EIDAS-Metadata
Encryption
Specific-Communication-v2
Specific-Communication-v2:: JCache-Ignite-Specific-Communication
EIDAS-Commons
EIDAS-Light-Commons

Figure 1: Dependencies between the installed modules

Legend

🟥 DO NOT USE
🟦 Recommended
🟦 Supported

3. Preparing the installation

This section provides instructions on how to deploy the project on Tomcat, Wildfly, GlassFish, WebLogic or WebSphere servers.

The appropriate JVM needs to be installed and configured first. If the selected application server includes an embedded JVM, the configuration still needs to be changed.

3.1. Configuring the JVM

The project is built by default using the **Java SDK** version **1.8** (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.

However recommended version is Java 7 update 141, Java 8 update 131 or above, since some eIDAS functionalities may not work, while using earlier versions of Java.

3.1.1. Oracle Java JCE Unlimited Strength Jurisdiction Policy

If Oracle provided JVM is going to be used, then it is necessary to 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 (subject to be moved by Oracle to different URI):
   - 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.

3.1.2. OpenJDK

Please be carefull when using OpenJDK, OpenJDK release content are not guaranteed. Therefore some compatibility problems may be encounterd while using OpenJDK with eIDAS.

3.1.3. IBM SDK Java

If the IBM provided JVM is going to be used for the eIDAS-Node, it is necessary to upgrade at least version 8.

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 (possibly subject to relocation by IBM).

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 (possibly subject to relocation by IBM).

3.1.3.1. Configuring encryption support

The default IBM 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
```

3.2. Configuring the application server

The following is a list of the supported servers.

Table 2: Supported servers

Application Server	Supported version(s)
Tomcat	8, 9
GlassFish	4.1 (web and full profile), 5 (full profile)
WildFly	11.0.0 (Java EE Full & Web Distribution) 15.0.1 (Java EE Full & Web

	Distribution)
WebLogic	12.1.2, 12.2.2c
WebSphere/WebSphere Liberty Profile	8.5.5 Liberty Profile Core 9

3.2.1. Configuring WebSphere

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

If your WAS installation is using IBM supplied Java SDK, please be sure to execute steps described in section 3.1.2.

3.2.2. Configuring WebSphere Liberty Profile

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 - possibly subject to being moved by IBM).

In order for eIDAS error messages to be properly displayed by WebSphere, add the property `<webContainer com.ibm.ws.webcontainer.enableErrorExceptionTypeFirst="true"/>` to the file `$SERVER_HOME/usr/servers/defaultServer/server.xml`.

The reason being that WebSphere deals with error page handling by first giving preference to HTTP error code and not to exceptions, which causes it to display an error page without the eIDAS error code /message.

Add this property in server.xml file of websphere:

<webContainer com.ibm.ws.webcontainer.redirectcontextroot="true"/>

If set to true, and a request is made to the context root of an application with a missing trailing slash, the WebContainer appends the trailing slash. The WebContainer redirects to the URL with the appended slash before it applies any servlet filters defined in the application.

3.3. Enabling logging

To enable audit logging of the communications between eIDAS-Node Proxy Service and eIDAS-Node Connector, please refer to the *eIDAS-Node Error and Event Logging* guide.

3.3.1. Configuring audit logging

Edit the project eIDAS-Node file: `logback.xml` (located in the resources 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>
```

3.3.2. Organisation of logs

The root level of logging defines the detail of logged events, for testing and development purposes, this level should be set to DEBUG. In the 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 the *eIDAS-Node Error and Event Logging* guide for more details):

- the Application System log (`eIDASNodeSystem`);
- the Application Security log (`eIDASNodeSecurity`);
- the Message Exchange log (`eIDASNodeSAMLExchange`) and
- the Application Detailed log (`eIDASNodeDetail`).

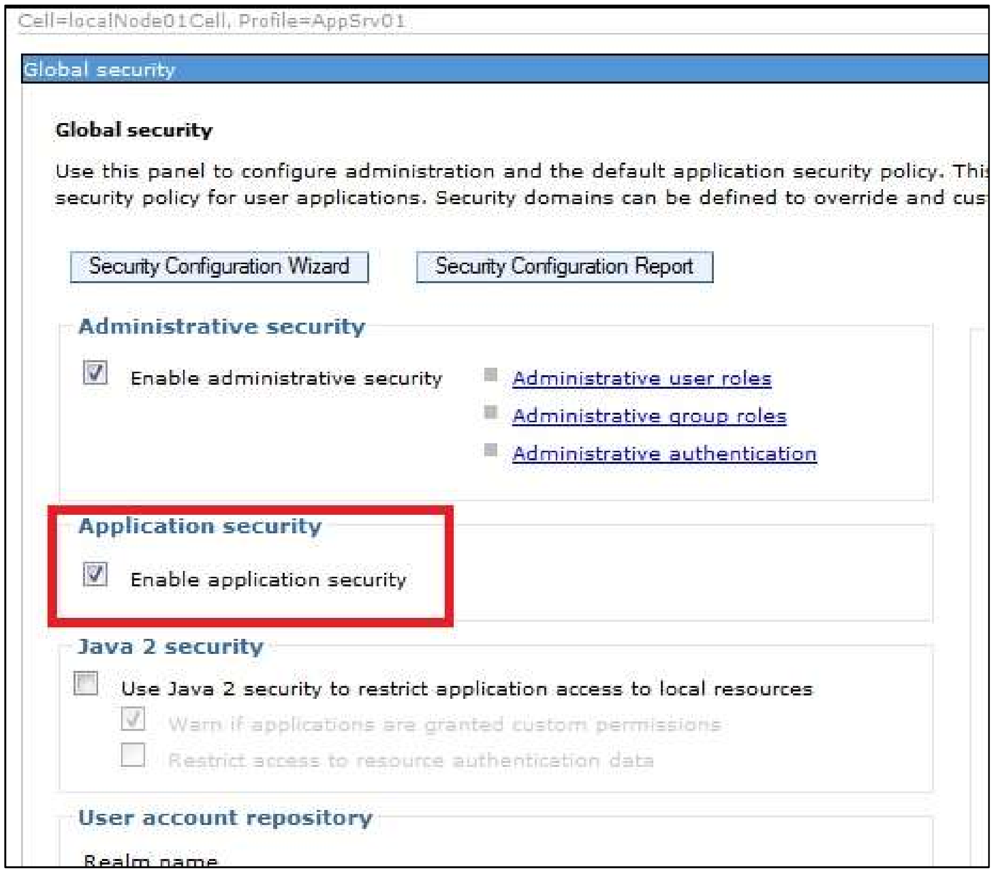
Set `${FILENAME_FULL_PATH}` for the location of the file which will contain the logs. (e.g.: `/opt/eidaslogs/eIDASNodeDetail.log`).

For further information on logging please refer to the *eIDAS-Node Error and Event Logging* and the *eIDAS-Node Security Considerations* guides.

3.4. Configuring application server security

3.4.1. 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.


Cell=localNode01Cell, Profile=AppSrv01

Global security

Global security

Use this panel to configure administration and the default application security policy. Thi
security policy for user applications. Security domains can be defined to override and cus

Administrative security

☑ Enable administrative security

- Administrative user roles
- Administrative group roles
- Administrative authentication

Application security

☑ Enable application security

Java 2 security

☐ Use Java 2 security to restrict application access to local resources

☑ Warn if applications are granted custom permissions

☐ Restrict access to resource authentication data

User account repository

Realm name

Figure 2: Enabling application security on WebSphere AS

4. Configuring the software

This section describes the configuration settings. Keep in mind that in production you need to enforce the configuration described in section 7.5 — *eIDAS-Node compliance*. Before proceeding with these steps your server must be configured, as described in section 3 — *Preparing the installation*.

Note: For information on implementing the eIDAS-Node Protocol Engine, please refer to the *CEF eID eIDAS-Node and SAML* document.

4.1. JVM system properties

In this section, are described the JVM system properties, that are used by or defined for the eIDAS-Node.

4.1.1. BouncyCastle security provider

When multiple applications dealing with Bouncycastle are deployed on the same application server, ClassCastException can be raised, when redeploying one of the application. This is because a JCE is loaded at global JVM level and not only in the application class loader and therefore, there is a discrepancy between the class loader of the JCE and its services. Therefore, the following system property is available:

-Dblock.security.provider.reinstall

to allow disable the reinstallation of JCE Providers for these types of applications. If block.security.provider.reinstall is missing or false, reinstallation of JCE providers can be triggered.

If block.security.provider.reinstall is true reinstallation of JCE providers is blocked.

4.1.2. Http forwarding through Proxy

The eIDAS Node has support for Http Proxy. In order for the proxy to be used, the following JVM properties can be used, depending on the needs:

- http.proxyHost
- http.proxyPort
- http.proxyUser
- http.proxyPassword
- http.nonProxyHosts

Note that the property http.nonProxyHosts could be needed to avoid using proxy for remote caches like ignite.

Example of proxy configuration without authentication:

```
set "JAVA_OPTS=%JAVA_OPTS% -Dhttp.proxyHost=192.168.137.129 -
Dhttp.proxyPort=8888 -Dhttp.nonProxyHosts=127.0.0.1"
```

4.2. Configuring the project

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

4.2.1. Setup configuration directory

The `$EIDAS_CONFIG_REPOSITORY` environment variable is used to locate the eIDAS-Node's directory of configuration files. It can be defined as an OS environment variable or by setting it to the runtime environment (by –D switch to JVM or on the AS admin console):

- $EIDAS_CONFIG_REPOSITORY – used in applicationContext.xml and points to the configuration directory of the application (e.g. file:/C:/PGM/projects/configEidas/).

By default `EIDAS_CONFIG_REPOSITORY` OS environment 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`. Please refer to `environmentalContext.xml` for more details on how to do it.

4.2.2. 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.

If the eIDAS-Node is configured to use encryption (essential in the production environment), 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.

For more information see the *CEF eID eIDAS-Node and SAML* manual.

4.2.3. Configuring with Basic Setup

The Basic Setup allows you to use predefined configuration supplied with the software package, only for demo purposes. Copy the provided configuration files to the predefined `EIDAS_CONFIG_REPOSITORY` and then 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
```

To configure the Demo Tools in order to test this Basic Setup, please read *eIDAS-Node Demo Tools Installation and Configuration Guide.*

4.3. eIDAS-Node configuration files

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

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

- General purpose parameters;
- eIDAS-Node Connector; and
- eIDAS-Node Proxy Service.

Also the `SamlEngine.xml` file contains the structure and properties to configure:

- SamlEngine for the Connector and the ProxyService
- Location in the EIDAS_REPOSITORY of the Configuration files respectively for Encryption, Signature, ProtocolProcessor, MetadataFetcher, internal components of a SamlEngine

Additionally, the eidas-node configuration repository has added in version 2.3 property files used to configure the metadata fetcher of the Connector and Service::

- server/metadata/MetadataFetcher_Connector.propertiesserver/metadata/MetadataFetcher_Service.properties

4.3.1. General purpose parameters

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

Table 3: General purpose parameters

Key	Description

Key	Description
`metadata.activate`	Allows activation/deactivation of SAML metadata (this parameter activates/deactivates metadata publishing and requesting on both Connector and Proxy Service (see also the *eIDAS-Node and SAML* manual*)*
`node.metadata.not.signed.descriptors`	List of URLs corresponding to entity descriptors whose signatures do not have to be checked. The format to use is http://descriptorurl1; https://descriptorurl2 etc.
`response.encryption.mandatory`	When set to 'true' the node encrypts assertions in the generated SAML responses (Note that the encryption related configuration must be in place). **Note**: this parameter is used by both Proxy Service and Connector nodes.
`disable.check.mandatory.eidas.attributes`	When set to 'false' the node will check if at least one set of mandatory attributes is included in the request or in the response. If set to 'true' there is no check. **Note:** this parameter is used by both Proxy Service and Connector nodes
`disable.check.representative.attributes`	When set to false, the ILightRequest is checked if there are Representative attributes requested, and reject the authentication request. Default is false.
`nonDistributedMetadata.retention`	Retention period for simple metadata cache in seconds. (Note: for distributed environment it's not used, set I up in hazelcastNode.xml instead)
`hashDigest.className`	Sets the digest class used by the IEidasLogger.
`metadata.file.repository`	Path to the static metadata files.
`metadata.http.retrieval`	Boolean value (true|false), which indicates whether the application will activate the use of the metadata from the HTTP URLs or use the static metadata.
`metadata.sector`	Value of the type of SP to be published in Connector's metadata, possible values: public and private.
`saml.connector`	Name of the configuration instance for the Connector's SAML Engine (defined in `SamlEngine.xml`).
`saml.service`	Name of the configuration instance for the Proxy Service's SAML Engine.
`response.sign.with.key.value`	When set to true, the eIDAS-Node marshals the signed Authentication Responses it originates to include the public RSA key instead of the full X509Certificate. When set to false the eIDAS-Node keeps the behaviour of 2.0.

Key	Description
`request.sign.with.key.value`	When set to true, the eIDAS-Node marshals the signed Authentication Requests it originates, to include the public RSA key instead of the full X509Certificate. When set to false the eIDAS-Node keeps the behaviour of 2.0.
`assertion.encrypt.with.key.value`	When set to true, the eIDAS-Node marshals the encrypted assertions added to the Authentication Responses it originates to include the public RSA key instead of the full X509Certificate. Otherwise, when set to false the eIDAS-Node keeps the behaviour of 2.0.
`eidas.protocol.version`	Value of eIDAS protocol version followed by the node, e.g. "1.1". When not empty, the value will be published in the node's metadata URLs.
`eidas.application.identifier`	Value of eIDAS protocol's application identifier relative to the node's code and version number., e.g. "CEF:eIDAS-ref:2.1". When not empty, the value will be published in the node's metadata URLs.
`include.assertion.fail.response.application.identifiers`	Values of the protocol versioning's application identifiers published in the metadata of the sender of the request, that need to receive not successful responses with an assertion. The values are comma or semicolon separated. For example, a possible value may be CEF:eIDAS-ref:1.4.2;CEF:eIDAS-ref:2.1, which will result in not successful response sent to CEF:eIDAS-ref:1.4.2 or CEF:eIDAS-ref:2.1 requesters will contain an assertion as before. Other cases will not. This property was introduced for interoperability reasons. It is to be used with eIDAS-Node versions 1.4.2 and below in the 1.4.x branch, and with eIDAS-Node versions 2.1 and below in the 2.x branch. Please note that this property will eventually disappear in the future once transition period ends and not successful responses with an assertion are no longer needed.

4.3.2. 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 and hard coded lists of supported core attributes in `LegalPersonSpec`, `NaturalPersonSpec`, `RepresentativeLegalPersonSpec`, and `RepresentativeNaturalPersonSpec` collected together in `EidasSpec` class, found in the SAMLEngine module.

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.

4.3.2.1. Attribute registry validation and metadata support

Besides the Attribute Registry XML files there is a hard coded list of supported core attributes in `LegalPersonSpec`, `NaturalPersonSpec`, `RepresentativeLegalPersonSpec`, and `RepresentativeNaturalPersonSpec` collected together in `EidasSpec` class, can be found in the SAMLEngine 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 of the eIDAS-Node Proxy Service.

4.3.3. eIDAS-Node Connector configuration

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

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

4.3.3.1. Service Provider configuration

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

Table 4: eIDAS-Node Connector and SP validation

Key	Description
`active.module.connector`	By setting this property to false, eIDAS-Node Connector will block the incoming Light Request and it will answer with an error message, also Connector's metadata will not be published, and Connector will not process any Proxy-service responses. The default value is true.

4.3.3.2. 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

Key	Description
`connector.assertion.url`	URL of the Action to be called when returning from eIDAS-Node Proxy Service. (This used as `AssertionConsumerServiceURL` in the Request also)
`saml.connector`	Name of the SAML ProtocolEngine instance used by the eIDAS-Node Connector in the eIDAS Network (between Connector and Proxy Service).
`connector.contact.support.email`	Email address of the support contact (for metadata)
`connector.contact.support.company`	Company name of the support contact (for metadata)
`connector.contact.support.givenname`	Given name of the support contact (for metadata)
`connector.contact.support.surname`	Surname of the support contact (for metadata)
`connector.contact.support.phone`	Phone number of the support contact (for metadata)
`connector.contact.technical.email`	Email address of the `technical` contact (for metadata)
`connector.contact.technical.company`	Company of the `technical` contact (for metadata)
`connector.contact.technical.givenname`	Given name of the technical contact (for metadata)
`connector.contact.technical.surname`	Surname of the technical contact (for metadata)
`connector.contact.technical.phone`	Phone number of the technical contact (for metadata)
`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, but does not set or validate the physical listener binding, therefore can be a custom value, like a reverse proxy external URL.
`connector.organization.name`	Name of the organization displayed in metadata
`connector.organization.displayname`	Localised display name of the organization for metadata
`connector.organization.url`	URL of the organisation for metadata containing information
`specific.connector.response.receiver`	URL for Specific Connector response receiver used when Specific Connector is built/deployed as WAR https://<specific ProxyService.*yourHostname*>:<specific ProxyService.*yourPort*>/SpecificProxyService/ ConnectorResponse
`connector.url.redirect.location.whitelist`	A list of urls to be checked against metadata url when redirect is used
`connector.url.post.location.whitelist`	A list of urls to be checked against metadata url when POST is used

Key	Description
`metadata.location.whitelist`	Semicolon separated urls of the ServiceProxies that the Connector is allowed to connect to obtain metadata/certificate to verify signature of SAML requests. This property is located in metadata/MetadataFetcher_Connector.properties Non-existent or empty whitelist will prevent the node from retrieving any metadata from any ServiceProxy.
`metadata.location.whitelist.use`	located in metadata/MetadataFetcher_Connector.properties This property is located in metadata/MetadataFetcher_Connector.properties This flag (default true) activates (or de-activates when false) the issuer metadata url whitelist logic. Non-existent parameter will default internally to true.

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

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

4.3.3.3. Configuring the recognised eIDAS-Node Proxy Service

The eIDAS-Node Connector recognises the eIDAS-Node Proxy Services listed in eidas.xml. Increment the `service.number`, 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 to Connector

Key	Description
`service.number`	Number of known eIDAS-Node Proxy Service
`serviceX.id`	Id of the eIDAS-Node Proxy Service X(=unique positive integer)
`serviceX.name`	Name of the eIDAS-Node Proxy Service X(=unique positive integer)
`serviceX.metadata.url`	URL where the eIDAS-Node Proxy Service X publishes its metadata. The collection of all these keys (i.e. service1.metadata.url … service8.metadata.url) is the whitelist of the proxy metadata url's that a Connector is allowed to connect to obtain metadata/certificate to verify signature of SAML responses.
`serviceX.skew.notbefore`	Time skew in milliseconds to adjust `notBefore` SAML condition in Connector. The actual value is added to the received time condition, negative value is possible.

Key	Description
`serviceX.skew.notonorafter`	Time skew in milliseconds to adjust `notOnOrAfter` SAML condition in Connector. The actual value is added to the received time condition. A negative value is possible.

4.3.4. eIDAS-Node Proxy Service configuration

To activate an eIDAS-Node Proxy Service the following properties need to be provided:

Table 7 : eIDAS-Node Proxy Service setup

Key	Description
`service.id`	NOT USED
`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. Used when the eIDAS-Node Proxy Service constructs the unique identifier attributes.
`service.contact.support.email`	Email address of the support contact (for metadata)
`service.contact.support.company`	Company of the support contact (for metadata)
`service.contact.support.givenname`	Given name of the support contact (for metadata)
`service.contact.support.surname`	Surname of the support contact (for metadata)
`service.contact.support.phone`	Phone number of the support contact (for metadata)
`service.contact.technical.email`	Email address of the `technical` contact (for metadata)
`service.contact.technical.company`	Company name of the `technical` contact (for metadata)
`service.contact.technical.givenname`	Given name of the technical contact (for metadata)
`service.contact.technical.surname`	Surname of the technical contact (for metadata)
`service.contact.technical.phone`	Phone number of the technical contact (for metadata)
`service.organization.name`	Name of the organisation displayed in the metadata
`service.organization.displayname`	Localised display name of the organisation for metadata
`service.organization.url`	URL of the organisation for Metadata containing information
`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 eIDAS-Node Proxy Service sends, but does not set or validate the physical listener binding, therefore can be a custom value, like a reverse proxy external URL.
`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 Checked against the Request.

Key	Description
`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 Does not come with physical binding check, so it can be set up for a reverse proxy external endpoint.
`ssos.serviceMetadataGeneratorIDP.post.location`	The URIfor 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 Does not come with physical binding check, so it can be set up for a reverse proxy external endpoint.
`metadata.location.whitelist`	Semicolon separated urls of the Connectors that the ServiceProxy is allowed to connect to obtain metadata/certificate to verify signature of SAML responses. This property is located in metadata/MetadataFetcher_Service.properties Non-existent or empty whitelist will prevent the node from retrieving any metadata from any Connector.
`metadata.location.whitelist.use`	This parameter and its value is located in SamlEngine.xml file under `<instance name=" Service"> element / <configuration name="ProtocolProcessorConf"> sub-element.` This property is located in metadata/MetadataFetcher_Service.properties Non-existent parameter will default internally to true.
`specific.proxyservice.request.receiver`	URL for Specific ProxyService requests receiver only used when Specific ProxyService is built/deployed as WAR https://<specific ProxyService.yourHostname>:<specific ProxyService.yourPort>/SpecificProxyService/ProxyServiceRequest
`replace.citizenCountryCode.by.spCountryCode.in.proxyService.lightRequest`	Boolean for activating the replace of Citizen Country Code value by the Service Provider Country Code Value in the Light Request send by the Proxy-Service to the Specific Proxy Service. If false or not set in eidas.xml the replacement of value will not be executed.
`insert.prefix.identifiers.country.code`	True: enable or disable the prefixing of identifiers with country codes in identifier attribute
`validate.prefix.country.code.identifiers`	True : enable or disable validation of prefixing identifier like attribute values
`enable.address.attribute.subject.confirmation.data`	False: enable or disable adding ipAddress to SubjectConfirmationData for SAML response assertion

4.3.4.1. eIDAS-Node Proxy Service activation/deactivation

Table 8: Activating the Proxy Service

Key	Description
`active.module.service`	By setting this property to false, eIDAS-Node Proxy-Service will block the eIDAS SAML request and it will answer with an error message, also Proxy-Service's metadata will not be published and any incoming SpecificProxyService Responses are blocked. The default value is true.

4.3.4.2. 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.
Please refer to Table 6: Adding eIDAS-Node Proxy Service to Connector for more information.

4.3.5. Additional configuration — Security

This section describes several configuration entries related to security policies. For more information about the security features please refer to the eIDAS-Node Security Considerations guide.

Table 9: Security policies

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

Table 10: Security HTTP header parameters

Key	Description
`security.header.CSP.enabled`	Enable/disable sending the Content Security Policy (CSP) header. CSP protects against the injection of foreign content.
`security.header.CSP. includeMozillaDirectives`	In the CSP, this additional directive can be added for backward compatibility with old Mozilla browsers.
`security.header. XXssProtection.block`	This header enables the cross-site-scripting (XSS) filter built into most recent web browsers.
`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.
`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'
`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.

Key	Description
`security.header.CSP.fallbackCheckMode`	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 a flood of warning messages logged by CSP report servlet. Disabled by default. When not set, the default false value will be applied.
`security.header.CSP.report.uri`	Prefix for **report_uri** header populated by the node in order to avoid retrieving the host/name from the request itself, that otherwise would have exposed the eidas flow to threats known as 'Host header poisoning' The eidas node populated this header with a value similar to http://eidasnode:8888/EidasNode/cspReportHandler where `security.header.CSP.report.uri = http://eidasnode:8888/EidasNode` At runtime, the uri above is compared with a uri build from the request. If the 2 values do not match, the report-uri directive will not be included in the CSP header.

Table 11: Check on certificate security parameter

Key	Description
`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.

4.3.5.1. Encryption

Table 12: Configuring encryption algorithm

Key	Description

`data.encryption.algorithm`	This is an override setting for values set in SAMLEngine configuration. 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;

Key	Description

`encryption.algorithm.whitelist`	This is an override setting for values set in SAMLEngine configuration. 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;

`check_certificate_validity_period`	Boolean value (true|false), which indicates if the application will disallow the use of obsolete certificates. Applies to the signature check also (see Table 13).
`disallow_self_signed_certificate`	Boolean value (true|false), which indicates if the application will disallow of the use of self-signed certificates. Applies to the signature check also (see Table 13).
`response.encryption.mandatory`	Boolean value (true/false), which indicates if the application will force the encryption of the SAML Response.

Note that additional configurations regarding the encryption can be done within the SamlEngine configuration (SamlEngine.xml) or in configuration files described/specify inside an instance declaration. For more information, please refer to the *CEF eID eIDAS-Node and SAML* manual.

4.3.5.2. Signature

Table 13: Signature algorithm

Key	Description
`signature.algorithm`	This is an override setting for values set in SAMLEngine configuration. 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 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 http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1 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.
`signature.algorithm.whitelist`	This is an override setting for values set in SAMLEngine configuration. 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 http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1
`response.sign.assertions`	When set to true, the SAML Responses (generated in Proxy Service and Connector) will have the attribute assertion signed

4.3.5.3. SAML Binding method

Table 14: SAML binding parameters

Key	Description
`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.
`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.

4.3.5.4. 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` and `eidasKeyStore_METADATA.jks` files, please see the *eIDAS-Node and SAML* manual for more information.

4.3.5.5. 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).

By default there is one distributed caches instance used by the Node for both correlation and anti-replay map purposes. This is loaded through the build depending on which EIDAS-JCache-XXXX-Node module is added as a dependency.

If EIDAS-JCache-Ignite-Node module/dependency is used for distributed caches at the node, jCacheImplNodeBeans.xml file contained in that module will provide the caches implementations based on Ignite.

The beans at the jCacheImplNodeBeans.xml file will provide the implementations of the caches used by the node.

```
<!-- production environment ignite initializer bean - injected into map
providers -->
<bean id="eidasIgniteInstanceInitializerNode"
class="eu.eidas.auth.cache.IgniteInstanceInitializerNode"
         init-method="initializeInstance" lazy-init="true">
        <property name="configFileName"
value="#{eidasConfigRepository}igniteNode.xml"/>
    </bean>
```

Figure 3: Node's Ignite instance name

If the EIDAS-JCache-Hazelcast-Node module/dependency is used, the jCacheImplNodeBeans.xml file contained in that module will provide the caches implementations based on Hazelcast.

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

Figure 4: Node's Hazelcast instance name

The Ignite instance is provided by the `eidasIgniteInstanceInitializerNode` bean while if for the case of Hazelcast the instance is provided by the `eidasHazelcastInstanceInitializer` bean.

```
<!-- production environment ignite initializer bean - injected into cache
providers -->
    <bean id="eidasIgniteInstanceInitializerNode"
class="eu.eidas.auth.cache.IgniteInstanceInitializerNode"
          init-method="initializeInstance" lazy-init="true">
        <property name="configFileName"
value="#{eidasConfigRepository}igniteNode.xml"/>
    </bean>
```

Figure 5: Node's Ignite instance provider bean

```
<!-- hazelcast initializer bean - injected into map providers -->
    <bean id="eidasNodeHazelcastInstanceInitializer"
class="eu.eidas.auth.cache.HazelcastInstanceInitializer"
          init-method="initializeInstance" lazy-init="true">
        <property name="hazelcastConfigfileName"
value="#{eidasConfigRepository}hazelcastNode.xml"/>
        <property name="hazelcastInstanceName"
ref="eidasNodeHazelcastInstance"/>
    </bean>
```

Figure 6: Node's Hazelcast instance provider bean

Note that #{eidasConfigRepository} value will originate from src/resources/environmentContext.xml.

This bean is injected into beans that have defined as class `ConcurrentMapServiceDistributedImpl` or `DistributedMetadataCaching`. If the distributed environment requires setup of multiple Hazelcast instances, the configuration can be done simply adding more of the above beans to `applicationContext`.

```
    <bean id="springServiceCMapAntiReplayProviderImpl"
class="eu.eidas.auth.cache.ConcurrentCacheServiceIgniteNodeImpl"
          lazy-init="true">
        <property name="igniteInstanceInitializer"
ref="eidasIgniteInstanceInitializerNode"/>
        <property name="cacheName" value="antiReplayCacheService"/>
    </bean>
    <bean id="springConnectorCMapAntiReplayProviderImpl"
          class="eu.eidas.auth.cache.ConcurrentCacheServiceIgniteNodeImpl" lazy-
init="true">
        <property name="igniteInstanceInitializer"
ref="eidasIgniteInstanceInitializerNode"/>
        <property name="cacheName" value="antiReplayCacheConnector"/>
    </bean>
```

Figure 7: Anti-replay cache configuration — Ignite — jCacheImplNodeBeans.xml (EIDAS-JCache-Ignite-Node module)

```
    <bean id="springServiceCMapAntiReplayProviderImpl"
class="eu.eidas.auth.cache.ConcurrentMapServiceDistributedImpl"
          lazy-init="true">
        <property name="hazelcastInstanceInitializer"
ref="eidasNodeHazelcastInstanceInitializer"/>
        <property name="cacheName" value="antiReplayCacheService"/>
    </bean>
    <bean id="springConnectorCMapAntiReplayProviderImpl"
class="eu.eidas.auth.cache.ConcurrentMapServiceDistributedImpl"
          lazy-init="true">
        <property name="hazelcastInstanceInitializer"
ref="eidasNodeHazelcastInstanceInitializer"/>
        <property name="cacheName" value="antiReplayCacheConnector"/>
    </bean>
```

Figure 8: Anti-replay cache configuration — Hazelcast — jCacheImplNodeBeans.xml (EIDAS-JCache-Hazelcast-Node module)

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

```
<!-- Correlation maps provided by Ignite for distributed environment, use these
in productions! -->
    <bean id="springConnectorCMapCorProviderImpl"
class="eu.eidas.auth.cache.ConcurrentCacheServiceIgniteNodeImpl"
          lazy-init="true">
        <property name="igniteInstanceInitializer"
ref="eidasIgniteInstanceInitializerNode"/>
        <property name="cacheName"
value="connectorRequestCorrelationCacheService"/>
    </bean>
    <bean id="springServiceCMapCorProviderImpl"
class="eu.eidas.auth.cache.ConcurrentCacheServiceIgniteNodeImpl"
          lazy-init="true">
        <property name="igniteInstanceInitializer"
ref="eidasIgniteInstanceInitializerNode"/>
        <property name="cacheName"
value="proxyServiceRequestCorrelationCacheService"/>
    </bean>
    <bean id="springConnectorCMapspecificLightCorProviderImpl"
          class="eu.eidas.auth.cache.ConcurrentCacheServiceIgniteNodeImpl" lazy-
init="true">
        <property name="igniteInstanceInitializer"
ref="eidasIgniteInstanceInitializerNode"/>
        <property name="cacheName"
value="specificConnectorLtRequestCorrelationCacheService"/>
    </bean>
```

Figure 9: Correlation caches configuration — Ignite — jCacheImplNodeBeans.xml (EIDAS-JCache-Ignite-Node module)

For more information about the Ignite product, please refer to Appendix C.

```
<bean id="springConnectorCMapCorProviderImpl"
class="eu.eidas.auth.cache.ConcurrentMapServiceDistributedImpl"
          lazy-init="true">
        <property name="hazelcastInstanceInitializer"
ref="eidasNodeHazelcastInstanceInitializer"/>
        <property name="cacheName"
value="connectorRequestCorrelationCacheService"/>
    </bean>
    <bean id="springServiceCMapCorProviderImpl"
class="eu.eidas.auth.cache.ConcurrentMapServiceDistributedImpl"
          lazy-init="true">
        <property name="hazelcastInstanceInitializer"
ref="eidasNodeHazelcastInstanceInitializer"/>
        <property name="cacheName"
value="proxyServiceRequestCorrelationCacheService"/>
    </bean>
    <bean id="springConnectorCMapspecificLightCorProviderImpl"
          class="eu.eidas.auth.cache.ConcurrentMapServiceDistributedImpl" lazy-
init="true">
        <property name="hazelcastInstanceInitializer"
ref="eidasNodeHazelcastInstanceInitializer"/>
        <property name="cacheName"
value="specificConnectorLtRequestCorrelationCacheService"/>
    </bean>
```

Figure 10: Correlation caches configuration — Hazelcast — jCacheImplNodeBeans.xml (EIDAS-JCache-Hazelcast-Node module)

For more information about the Hazelcast product, please refer to section 7.3 — *Set up Hazelcast* and Appendix D.

4.3.5.6. Error Codes and Error Messages

The full list of eIDAS-Node error codes and related error messages is shown in the *eIDAS-Node Error Codes* document. 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)

4.3.6. Specific properties

For the Basic Setup, you might need to reconfigure MS-Specific module Configuration for that application as detailed in the *eIDAS-Node Demo Tool Installation and Configuration Guide*.

4.3.7. Demo Service Provider

For the Basic Setup, you might need to reconfigure Demo Service Provider. Configuration for that application is detailed in the *eIDAS-Node Demo Tool Installation and Configuration Guide*.

4.3.8. Demo Identity Provider

In order to proceed with Basic Setup, you might need to modify the configuration of Demo Identity Provider. The procedure and settings are detailed in the *eIDAS-Node Demo Tool Installation and Configuration Guide*.

5. Building and deploying the software

This section describes the steps to build and then to deploy the software on the supported servers. There are two main types of eIDAS-Node: Connector and Proxy Service.

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 3.5.4 and above. Lower versions can result in exceptions.

There are two ways to build the binaries from sources:

1. **Parent build**: the `pom.xml` file in the EIDAS-Parent module is a common reference for all dependent module/external Maven artefact versions, and able to build all binaries related to EidasNode and/or Demo Tools.

   There are various profiles to help tailoring the build to one's particular needs: these can be split in two main categories.

   First: profiles related to application server specifics, for instance profiles named weblogic.

   Second: two profiles related to the scope of modules to be built, specifically NodeOnly (this is active by default,) and DemoToolsOnly.

   For instance issuing Maven "install" command with the appropriate activation profile (e.g. for WebLogic: `-P weblogic,NodeOnly,DemoTools`) will result in a full build.

   Third: several profiles are present in that switch between Jcache implementations for Node and for Specific Communication Caches. Those are profiles

   -PnodeJcacheIgnite : Ignite JCache implementation;

   -PnodeJcacheHazelcast: Hazelcast 3.2 JCache implementation/adaptation;

   -PnodeJcacheDev : Guava JCache implementation/adaptation (Only for non distributed case).

   -PspecificCommunicationJcacheIgnite : Ignite JCache implementation;

   -PspecificCommunicationJcacheHazelcast: Hazelcast 3.2 JCache implementation/adaptation;

   -PspecificCommunicationJcacheDev: Local Caches implementation (possible only for Monolithic deployment)

   -PspecificCommunicationJcacheProvidedImpl: to be provided Jcache implementation.

   Note that among these, the build has to be executed with one of the nodeXXX profiles and one for the specificCommunicationXXX profiles so that the node works properly.

2. **Module-based build**: it is possible to build the artefacts one-by-one, which can be helpful if there is a need to build just one module. In this case please remember the dependencies between them. There is a certain order that needs to be followed.

The next sections detail the above two methods for supported application servers.

5.1. Tomcat/GlassFish server deployment

You must compile, install and deploy the projects, either by compiling the parent 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.

- **Note:** $GLASSFISH_HOME refers to the base directory of your GlassFish server (e.g. /home/user/apps/glassfishv4).

Table 15: Parent project build for Tomcat/GlassFish Server deployment

Step	Folder	Command line

1	`EIDAS-Parent`	`mvn clean install –PNodeOnly[,DemoToolsOnly]`
		
		`[-P nodeJcacheIgnite, nodeJcacheHazelcast nodeJcacheProvidedImpl ]`
		
		`[-P specificCommunicationJcacheIgnite, specificCommunicationJcacheHazelcast, specificCommunicationJcacheDev, specificCommunicationJcacheProvidedImpl ]`
		
		`[-DspecificJar]`
		
		Note one of the nodeJcacheX profiles as well as one of specificCommunicationJcacheX profiles needs to be chosen.
		
		After the build has been done, deploy EidasNode.war, IdP.war, SP.war, SpecificConnector.war and SpecificProxyService.war .

Table 16: 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-JCache-Dev`	`mvn clean install`
5	`EIDAS-JCache-Dev-Node`	`mvn clean install`
6	`EIDAS-JCache-Dev-Specific-Communication`	`mvn clean install`
7	`EIDAS-JCache-Ignite`	`mvn clean install`

Step	Folder	Command line
8	EIDAS-JCache-Ignite-Node	mvn clean install
9	EIDAS-JCache-Ignite-Specific-Communication	mvn clean install
10	EIDAS-JCache-Hazelcast	mvn clean install
11	EIDAS-JCache-Hazelcast-Node	mvn clean install
12	EIDAS-JCache-Hazelcast-Specific-Communication	mvn clean install
13	EIDAS-SpecificCommunicationDefinition	mvn clean install [-P nodeJcacheIgnite, nodeJcacheHazelcast nodeJcacheProvidedImpl] [-P specificCommunicationJcacheIgnite, specificCommunicationJcacheHazelcast, specificCommunicationJcacheDev, specificCommunicationJcacheProvidedImpl] [-DspecificJar] Note one of the nodeJcacheX profiles as well as one of specificCommunicationJcacheX profiles needs to be chosen.
14	EIDAS-Encryption	mvn clean install
15	EIDAS-ConfigModule	mvn clean install
16	EIDAS-Metadata	mvn clean install
17	EIDAS-SAMLEngine	mvn clean install
18	EIDAS-Updater	mvn clean install
19	EIDAS-Node	a. mvn clean package b. Tomcat: copy target/EidasNode.war $TOMCAT_HOME/webapps/EidasNode.war GlassFish: copy target/EidasNode.war $GLASSFISH_DOMAIN/autodeploy/EidasNode.war

5.2. WildFly 11.0.0 and 15.0.1 Server deployment

You must compile, install and deploy the projects, either by compiling the parent 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 17: Parent project build for WildFly 11.0.0/WildFly 15.0.1 Server deployment

Step	Folder	Command line

1	`EIDAS-Parent`	`mvn clean install -PNodeOnly,DemoToolsOnly`
		
		`[-P nodeJcacheIgnite, nodeJcacheHazelcast nodeJcacheProvidedImpl ]`
		
		`[-P specificCommunicationJcacheIgnite, specificCommunicationJcacheHazelcast, specificCommunicationJcacheDev, specificCommunicationJcacheProvidedImpl ]`
		
		`[-DspecificJar]`
		
		`Note one of the nodeJcacheX profiles as well as one of specificCommunicationJcacheX profiles needs to be chosen.`
		
		After the build has been done, deploy `EidasNode.war`, `IdP.war`, `SP.war`, `SpecificConnector.war` and `SpecificProxyService.war` .

Table 18: Module-based build for WildFly 11.0.0/WildFly 15.0.1 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-JCache-Dev`	`mvn clean install`
5	`EIDAS-JCache-Dev-Node`	`mvn clean install`
6	`EIDAS-JCache-Dev-Specific-Communication`	`mvn clean install`
7	`EIDAS-JCache-Ignite`	`mvn clean install`
8	`EIDAS-JCache-Ignite-Node`	`mvn clean install`
9	`EIDAS-JCache-Ignite-Specific-Communication`	`mvn clean install`

Step	Folder	Command line
10	EIDAS-JCache-Hazelcast	`mvn clean install`
11	EIDAS-JCache-Hazelcast-Node	`mvn clean install`
12	EIDAS-JCache-Hazelcast-Specific-Communication	`mvn clean install`

13	EIDAS-SpecificCommunicationDefinition	`mvn clean install`
		
		`[-P nodeJcacheIgnite, nodeJcacheHazelcast nodeJcacheProvidedImpl ]`
		
		`[-P specificCommunicationJcacheIgnite, specificCommunicationJcacheHazelcast, specificCommunicationJcacheDev, specificCommunicationJcacheProvidedImpl ]`
		
		`[-DspecificJar]`
		
		Note one of the nodeJcacheX profiles as well as one of specificCommunicationJcacheX profiles needs to be chosen.

14	EIDAS-Encryption	`mvn clean install`
15	EIDAS-ConfigModule	`mvn clean install`
16	EIDAS-Metadata	`mvn clean install`
17	EIDAS-SAMLEngine	`mvn clean install`
18	EIDAS-Updater	`mvn clean install`

19	EIDAS-Node	`mvn clean package -P wildlfy`
		
		`copy target/EidasNode.war $WILDFLY_HOME/ standalone/deployments/EidasNode.war`

5.3. WebLogic Server deployment

You must compile, install and deploy the projects, either by compiling the parent 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 19: Parent project build for WebLogic Server deployment

Step	Folder	Command line
1	EIDAS-Parent	For Welogic 12.1: mvn clean install -P weblogic, NodeOnly,DemoToolsOnly -Dweblogic12.1.3-BouncyCastle For Welogic 12.2: mvn clean install -P weblogic,NodeOnly,DemoToolsOnly [-P nodeJcacheIgnite, nodeJcacheHazelcast nodeJcacheProvidedImpl] [-P specificCommunicationJcacheIgnite, specificCommunicationJcacheHazelcast, specificCommunicationJcacheDev, specificCommunicationJcacheProvidedImpl] [-DspecificJar] Note one of the nodeJcacheX profiles as well as one of specificCommunicationJcacheX profiles needs to be chosen. After the build has been done, deploy EidasNode.war, IdP.war, SP.war, SpecificConnector.war and SpecificProxyService.war .

Table 20: 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-JCache-Dev	mvn clean install
5	EIDAS-JCache-Dev-Node	mvn clean install
6	EIDAS-JCache-Dev-Specific-Communication	mvn clean install
7	EIDAS-JCache-Ignite	mvn clean install
8	EIDAS-JCache-Ignite-Node	mvn clean install
9	EIDAS-JCache-Ignite-Specific-Communication	mvn clean install

Step	Folder	Command line
10	`EIDAS-JCache-Hazelcast`	`mvn clean install`
11	`EIDAS-JCache-Hazelcast-Node`	`mvn clean install`
12	`EIDAS-JCache-Hazelcast-Specific-Communication`	`mvn clean install`

13	`EIDAS-SpecificCommunicationDefinition`	`mvn clean install`
		
		`[-P nodeJcacheIgnite, nodeJcacheHazelcast nodeJcacheProvidedImpl ]`
		
		`[-P specificCommunicationJcacheIgnite, specificCommunicationJcacheHazelcast, specificCommunicationJcacheDev, specificCommunicationJcacheProvidedImpl ]`
		
		`[-DspecificJar]`
		
		`Note one of the nodeJcacheX profiles as well as one of specificCommunicationJcacheX profiles needs to be chosen.`

14	`EIDAS-Encryption`	`mvn clean install`
15	`EIDAS-ConfigModule`	`mvn clean install`
16	`EIDAS-Metadata`	`mvn clean install`
17	`EIDAS-SAMLEngine`	`mvn clean install`
18	`EIDAS-Updater`	`mvn clean install`

19	`EIDAS-Node`	`a. mvn clean package -P weblogic`
		
		`b. copy target/EidasNode.war $WLS_HOME/DOMAIN/ autodeploy/EidasNode.war`

5.4. WebSphere Server deployment

You must compile, install and deploy the projects, either by compiling the parent 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 21: Parent project build for WebSphere Server deployment

Step	Folder	Command line

1	`EIDAS-Parent`	`mvn clean install -PNodeOnly,DemoToolsOnly`
		
		`[-P nodeJcacheIgnite, nodeJcacheHazelcast nodeJcacheProvidedImpl ]`
		
		`[-P specificCommunicationJcacheIgnite, specificCommunicationJcacheHazelcast, specificCommunicationJcacheDev, specificCommunicationJcacheProvidedImpl ]`
		
		`[-DspecificJar]`
		
		Note one of the nodeJcacheX profiles as well as one of specificCommunicationJcacheX profiles needs to be chosen.
		
		After the build has been done, deploy `EidasNode.war`, `IdP.war` and `SP.war`, `SpecificConnector.war` and `SpecificProxyService.war`.

Table 22: 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-JCache-Dev`	`mvn clean install`
5	`EIDAS-JCache-Dev-Node`	`mvn clean install`
6	`EIDAS-JCache-Dev-Specific-Communication`	`mvn clean install`
7	`EIDAS-JCache-Ignite`	`mvn clean install`
8	`EIDAS-JCache-Ignite-Node`	`mvn clean install`
9	`EIDAS-JCache-Ignite-Specific-Communication`	`mvn clean install`
10	`EIDAS-JCache-Hazelcast`	`mvn clean install`
11	`EIDAS-JCache-Hazelcast-Node`	`mvn clean install`
12	`EIDAS-JCache-Hazelcast-Specific-Communication`	`mvn clean install`

Step	Folder	Command line
13	EIDAS-SpecificCommunicationDefinition	mvn clean install [-P nodeJcacheIgnite, nodeJcacheHazelcast nodeJcacheProvidedImpl] [-P specificCommunicationJcacheIgnite, specificCommunicationJcacheHazelcast , specificCommunicationJcacheDev, specificCommunicationJcacheProvidedImpl] [-DspecificJar] Note one of the nodeJcacheX profiles as well as one of specificCommunicationJcacheX profiles needs to be chosen.
14	EIDAS-Encryption	mvn clean install
15	EIDAS-ConfigModule	mvn clean install
16	EIDAS-Metadata	mvn clean install
17	EIDAS-SAMLEngine	mvn clean install
18	EIDAS-Updater	mvn clean install
19	EIDAS-Node	mvn clean package

5.5. Monolithic Deployment

Besides the 'Basic Deployment' described in this document, a 'Monolithic Deployment' is possible. In this case the `EidasNode.war` will include `SpecificConnector` and `SpecificProxyService` modules as JARs.

In this case add `-D specificJar` to the build commands for the following modules:

- EIDAS-SpecificCommunicationDefinition
- EIDAS-Node

This also applies to Demo Tools modules, so please check the *Monolithic Deployment* section in the *Demo Tools Installation and Configuration Guide* for more details.

Lastly, if monolithic deployment will be performed, the operator will need to follow and take into consideration the document above (*Demo Tools Installation and Configuration Guide*), notably the configuration parameters such relaystate.randomize.null , etc.

In this case the, `EIDAS-JCache-Dev will be included by default as the Jcache implementation of the Communication Caches between Node and MS Specific parts.`

It is possible to change the Jcache implementation to Jcache-Ignite or Jcache-Hazelcast, by activating profile `specificCommunicationJcacheIgnite or specificCommunicationJcacheHazelcast`, respectively.

6. Verifying the installation

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 is also shown so you can verify that your applications were built successfully.

6.1. Tomcat 7, 8

```
$TOMCAT_HOME/webapps/
      EidasNode.war

      (server specific directories were not included)
```

6.2. WildFly 11.0.0

- Replace if not done before the files contained in ./modules/system/layers/base/org/bouncycastle/main/ files by the ones in /AdditionalFiles/Wildfly11/org/bouncycastle/main/ (Note the bcprov-jdk15on-1.64.jar is a signed version obtained from https://www.bouncycastle.org/download/bcprov-jdk15on-164.jar)
- Add the following line in %WILDFLY_HOME%\modules\system\layers\base\sun\jdk\main\module.xml in the <paths> element

  ```
  <path name="com/sun/org/apache/xalan/internal/xsltc/trax"/>
  ```

- Copy war files under $WILDFLY_HOME/standalone/Deployments.

6.3. WildFly 15.0.1

- Replace if not done before the files contained in ./modules/system/layers/base/org/bouncycastle/main/ files by the ones in /AdditionalFiles/Wildfly15/org/bouncycastle/main/ (Note the bcprov-jdk15on-1.64.jar is a signed version obtained from https://www.bouncycastle.org/download/bcprov-jdk15on-164.jar)
- Copy war files under $WILDFLY_HOME/standalone/Deployments.

6.4. GlassFish V4.1, V5

6.4.1. GlassFish V4.1

```
$GLASSFISH_DOMAIN/autodeploy/
      EidasNode.war

      (server specific directories were not included)
```

6.4.2. GlassFish V5

```
$GLASSFISH_DOMAIN/autodeploy/
      EidasNode.war

      (server specific directories were not included)
```

6.5. WebLogic

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

6.6. 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.

In **Enterprise Applications > EidasNode > ClassLoader** choose:

- **Class loader order** to: Classes loaded with local class loader first (parent last);
- **WAR class loader policy** to: Single class loader for application

Note: for WebSphere Liberty Profile deployment see section 3.2.2 — *Configuring WebSphere Liberty Profile*.

6.7. Configuration files

The below configuration and keystore files are needed for the installation of the eIDAS-Node. The layout itself can be different, depending on the environment variables, so this is just an example of Basic Setup:

```
server/eidas.xml
server/encryptionConf.xml
server/EncryptModule_Connector.xml
server/EncryptModule_Service.xml
server/hazelcastNode.xml(needed only if profile nodeJcacheHazelcast is
activated )
server/hazelcastSpecificCommunication.xml (needed only if profile
specificCommunicationJcacheHazelcast is activated )
server/igniteNode.xml(needed only in default build or if profile
nodeJcacheIgnite is activated )
server/igniteSpecificCommunication.xml(needed only in default build or
if profile specificCommunicationJcacheIgnite is activated )
server/saml-engine-additional-attributes.xml
server/SamlEngine.xml
server/SamlEngine_Connector.xml
server/SamlEngine_Service.xml
server/SignModule_Connector.xml
server/SignModule_Service.xml
server/specificConnector/specificCommunicationDefinitionConnector.xml
server/specificProxyService/specificCommunicationDefinitionProxyservice.xml
keystore/eidasKeyStore.jks
keystore/eidasKeyStore_Connector_CA.jks
keystore/eidasKeyStore_Connector_CB.jks
keystore/eidasKeyStore_Connector_CC.jks
```

```
keystore/eidasKeyStore_Connector_CD.jks
keystore/eidasKeyStore_Connector_CF.jks
keystore/eidasKeyStore_METADATA.jks
keystore/eidasKeyStore_Service_CA.jks
keystore/eidasKeyStore_Service_CB.jks
keystore/eidasKeyStore_Service_CC.jks
keystore/eidasKeyStore_Service_CD.jks
keystore/eidasKeyStore_Service_CF.jks
```

7. Advanced configuration for production environments

This section provides detailed descriptions of the configurations to enable you to change specific aspects as required.

7.1. Clustering environment

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

7.1.1. 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.

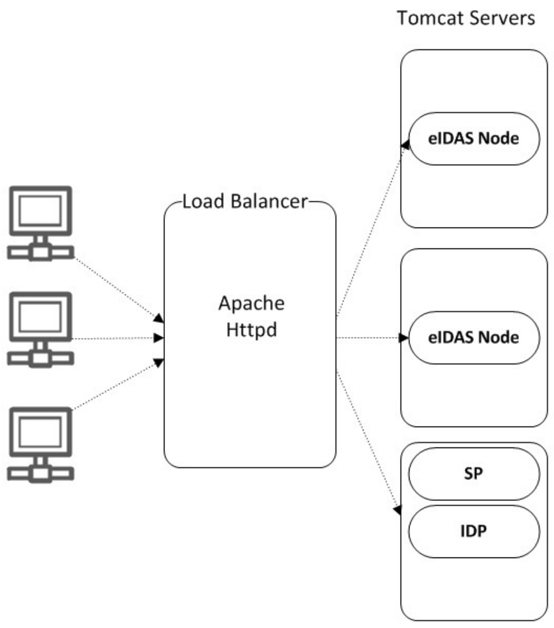


Figure 11: Clustering environment — Load balancer

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.

7.1.2. Load balancer with Hazelcast

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

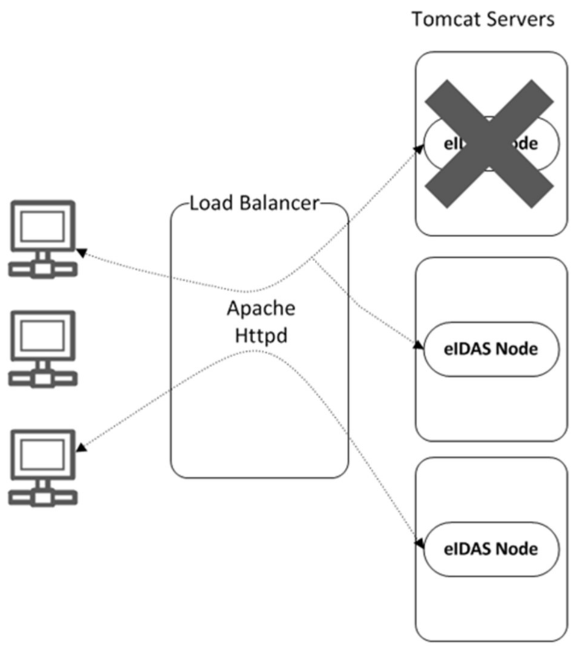


Figure 12: Clustering environment — Load Balancer with Hazelcast

For Hazelcast, replication of message exchange states (in correlation maps) needs to be set up (see section 7.3 — *Set up Hazelcast*).

7.2. Configuring Tomcat

7.2.1. 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

7.2.2. 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**.

7.2.2.1. 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
```

7.2.2.2. 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:

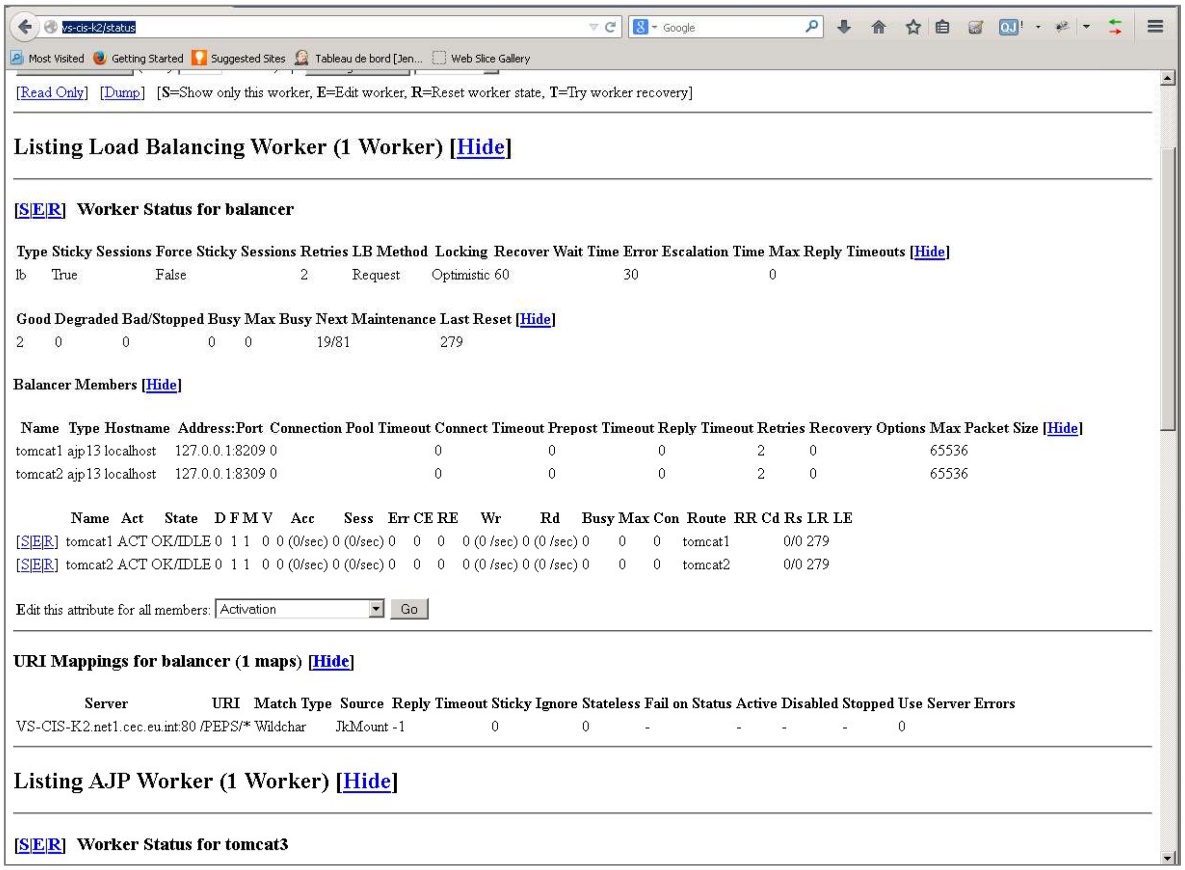
```
worker.list=balancer,stat,tomcat3
worker.tomcat1.type=ajp13
worker.tomcat1.port=8209
worker.tomcat1.host=localhost
worker.tomcat2.type=ajp13
worker.tomcat2.port=8309
worker.tomcat2.host=localhost
worker.tomcat3.type=ajp13
worker.tomcat3.port=8409
worker.tomcat3.host=localhost
worker.balancer.type=lb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

7.3. Set up Hazelcast

To replicate required information between cluster members, all nodes need to be configured with Hazelcast. Please refer to section 4.3.5.5 — *Additional Configuration — Anti-replay Cache and Correlation Map Configuration* and Appendix C for information on how to implement the required configuration.

7.4. Check your installation

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



vs-cis-k2/status

[Read Only] [Dump] [S=Show only this worker, E=Edit worker, R=Reset worker state, T=Try worker recovery]

Listing Load Balancing Worker (1 Worker) [Hide]

[S|E|R] Worker Status for balancer

Type	Sticky Sessions	Force Sticky Sessions	Retries	LB Method	Locking	Recover Wait Time	Error Escalation Time	Max Reply Timeouts [Hide]
lb	True	False	2	Request	Optimistic	60	30	0

Good	Degraded	Bad/Stopped	Busy	Max Busy	Next Maintenance	Last Reset [Hide]
2	0	0	0	0	19/81	279

Balancer Members [Hide]

Name	Type	Hostname	Address:Port	Connection Pool Timeout	Connect Timeout	Prepost Timeout	Reply Timeout	Retries	Recovery Options	Max Packet Size [Hide]
tomcat1	ajp13	localhost	127.0.0.1:8209	0	0	0	0	2	0	65536
tomcat2	ajp13	localhost	127.0.0.1:8309	0	0	0	0	2	0	65536

	Name	Act	State	D	F	M	V	Acc	Sess	Err	CE	RE	Wr	Rd	Busy	Max	Con	Route	RR	Cd	Rs	LR	LE
[S|E|R]	tomcat1	ACT	OK/IDLE	0	1	1	0	0 (0/sec)	0 (0/sec)	0	0	0	0 (0 /sec)	0 (0 /sec)	0	0	0	tomcat1			0/0	279	
[S|E|R]	tomcat2	ACT	OK/IDLE	0	1	1	0	0 (0/sec)	0 (0/sec)	0	0	0	0 (0 /sec)	0 (0 /sec)	0	0	0	tomcat2			0/0	279	

Edit this attribute for all members: Activation Go

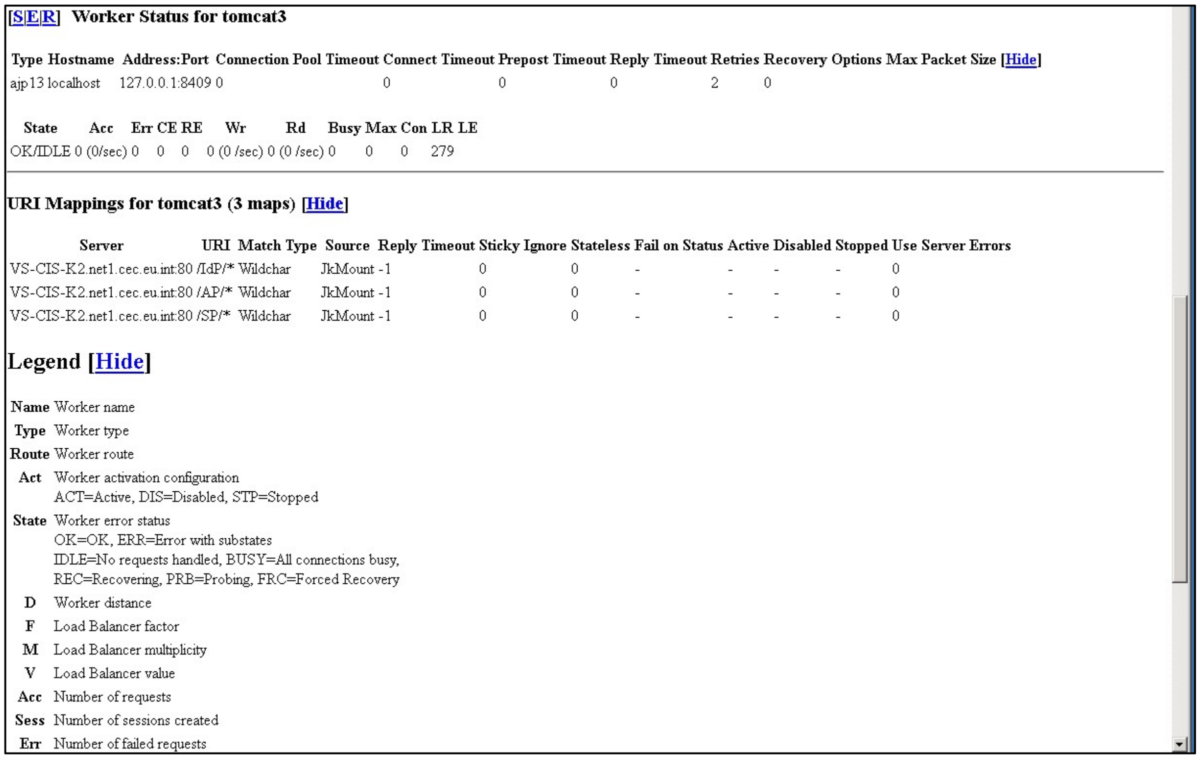
URI Mappings for balancer (1 maps) [Hide]

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	/PEPS/*	Wildchar	JkMount	-1	0	0	-	-	-	-	0

Listing AJP Worker (1 Worker) [Hide]

[S|E|R] Worker Status for tomcat3

Figure 13: 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 14: Apache status page (continued)

7.5. eIDAS-Node compliance

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

Table 23: 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

Parameter	Resulting value
`tls.enabled.ciphers`	The eIDAS supported cipher suites have been consolidated according to the eIDAS-Crypto requirements. The eidas.xml file has been adapted and lists all the supported cipher suites for Java7 and Java8. For authorized persons more details can be found at https://ec.europa.eu/cefdigital/wiki/x/6MXuAw Supported cipher suites for java 7 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, TLS_EMPTY_RENEGOTIATION_INFO_SCSV Supported cipher suites for java 8 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256, TLS_DHE_RSA_WITH_AES_128_GCM_SHA256, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256, TLS_DHE_RSA_WITH_AES_256_GCM_SHA384, TLS_EMPTY_RENEGOTIATION_INFO_SCSV
`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)
`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,

Parameter	Resulting value
`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 existence of Representative attributes in requests. Note: this parameter is used by both eIDAS-Node Proxy Service and eIDAS-Node 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 as the citizen country of the SamlRequest

Note that to ensure compliance, the following checks are also made by the code and are not parametrized:

- the Level of Assurance indicated in the Assertion matches or exceeds the requested Level of Assurance (see Appendix A); and
- 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 C for further details)

Appendix A. 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 B. 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 C. Ignite proposed configuration

For the communication caches, an Ignite implementation can be used. The configuration of the product is done via its configuration file ignite.xml located by EIDAS_CONFIG_REPOSITORY. A default configuration is provided with the application, e.g. in igniteNode.xml file, we can see a simple configuration for the node using Ignite Jcache as copied below. Note that, for more support on Ignite configuration, the Ignite Documentation should be used.

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



<!--
  ~ Copyright (c) 2018 by European Commission
  ~
  ~ Licensed under the EUPL, Version 1.2 or - as soon they will be
  ~ approved by the European Commission - subsequent versions of the
  ~ EUPL (the "Licence");
  ~ You may not use this work except in compliance with the Licence.
  ~ You may obtain a copy of the Licence at:
  ~ https://joinup.ec.europa.eu/page/eupl-text-11-12
  ~
  ~ Unless required by applicable law or agreed to in writing, software
  ~ distributed under the Licence is distributed on an "AS IS" basis,
  ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
  ~ implied.
  ~ See the Licence for the specific language governing permissions and
  ~ limitations under the Licence.
  -->

<!--
    Ignite Spring configuration file to startup Ignite cache.

    This file demonstrates how to configure cache using Spring. Provided cache
    will be created on node startup.

    Use this configuration file when running HTTP REST examples (see
'examples/rest' folder).

    When starting a standalone node, you need to execute the following command:
    {IGNITE_HOME}/bin/ignite.{bat|sh} examples/config/ignite-cache.xml

    When starting Ignite from Java IDE, pass path to this file to Ignition:
    Ignition.start("examples/config/ignite-cache.xml");
-->


<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.xsd">

    <bean id="igniteNode.cfg"
class="org.apache.ignite.configuration.IgniteConfiguration">

        <property name="igniteInstanceName" value="igniteNodeTOM"/>

        <property name="cacheConfiguration">
            <list>
                <!-- Partitioned cache example configuration (Atomic mode). -->
                <bean class="org.apache.ignite.configuration.CacheConfiguration">
                    <property name="name" value="antiReplayCacheService"/>
                    <property name="atomicityMode" value="ATOMIC"/>
                    <property name="backups" value="1"/>
                    <property name="expiryPolicyFactory"
                              ref="3_hours_duration"/>
                </bean>
                <!-- Partitioned cache example configuration (Atomic mode). -->
                <bean class="org.apache.ignite.configuration.CacheConfiguration">
                    <property name="name" value="antiReplayCacheConnector"/>
                    <property name="atomicityMode" value="ATOMIC"/>
                    <property name="backups" value="1"/>
                    <property name="expiryPolicyFactory"
                              ref="3_hours_duration"/>
                </bean>
                <!-- Partitioned cache example configuration (Atomic mode). -->
                <bean class="org.apache.ignite.configuration.CacheConfiguration">
                    <property name="name"
value="connectorRequestCorrelationCacheService"/>
                    <property name="atomicityMode" value="ATOMIC"/>
                    <property name="backups" value="1"/>
                    <property name="expiryPolicyFactory"
                              ref="7_minutes_duration"/>
                </bean>
                <!-- Partitioned cache example configuration (Atomic mode). -->
                <bean class="org.apache.ignite.configuration.CacheConfiguration">
                    <property name="name"
value="proxyServiceRequestCorrelationCacheService"/>
                    <property name="atomicityMode" value="ATOMIC"/>
                    <property name="backups" value="1"/>
                    <property name="expiryPolicyFactory"
                              ref="7_minutes_duration"/>
                </bean>
                <!-- Partitioned cache example configuration (Atomic mode). -->
                <bean class="org.apache.ignite.configuration.CacheConfiguration">
                    <property name="name"
value="specificConnectorLtRequestCorrelationCacheService"/>
                    <property name="atomicityMode" value="ATOMIC"/>
                    <property name="backups" value="1"/>
                    <property name="expiryPolicyFactory"
                              ref="7_minutes_duration"/>
                </bean>
                <bean class="org.apache.ignite.configuration.CacheConfiguration">
                    <property name="name" value="connectorFlowIdCacheService"/>
                    <property name="atomicityMode" value="ATOMIC"/>
                    <property name="backups" value="1"/>
                    <property name="expiryPolicyFactory"
                              ref="7_minutes_duration"/>
                </bean>
                <bean class="org.apache.ignite.configuration.CacheConfiguration">
                    <property name="name"
value="proxyServiceFlowIdCacheService"/>
                    <property name="atomicityMode" value="ATOMIC"/>
                    <property name="backups" value="1"/>
                    <property name="expiryPolicyFactory"
                              ref="7_minutes_duration"/>
                </bean>
                <!-- Partitioned cache example configuration (Atomic mode). -->
                <bean class="org.apache.ignite.configuration.CacheConfiguration">
                    <property name="name" value="eidasmetadata"/>
                    <property name="atomicityMode" value="ATOMIC"/>
                    <property name="backups" value="1"/>
                    <property name="expiryPolicyFactory"
                              ref="24_hours_duration"/>
                </bean>
            </list>
        </property>

        <!--Multicast discover of other nodes in the grid configuration-->
        <property name="discoverySpi">
            <bean class="org.apache.ignite.spi.discovery.tcp.TcpDiscoverySpi">
                <property name="ipFinder">
                    <bean
class="org.apache.ignite.spi.discovery.tcp.ipfinder.multicast.TcpDiscoveryMultica
stIpFinder">
```

```
                        <property name="multicastGroup" value="228.10.10.165"/>
                    </bean>
                </property>
            </bean>
        </property>

        <!-- Ssl/Tls context. -->
        <!--IMPORTANT: THIS IS A DEMO CONFIGURATION AND DEMO KEYSTORES, IT NEEDS
TO BE CHANGED FOR PRODUCTION ALSO THE KEYS IN THE KEYSTORES NEED TO BE CREATED AS
WELL-->
        <property name="sslContextFactory">
            <bean class="org.apache.ignite.ssl.SslContextFactory">
                <property name="keyStoreFilePath"
value="${EIDAS_CONFIG_REPOSITORY}/ignite/KeyStore/server.jks"/>
                <property name="keyStorePassword" value="123456"/>
                <property name="trustStoreFilePath"
value="${EIDAS_CONFIG_REPOSITORY}/ignite/KeyStore/trust.jks"/>
                <property name="trustStorePassword" value="123456"/>
                <property name="protocol" value="TLSv1.2"/>
            </bean>
        </property>

        <!-- how frequently Ignite will output basic node metrics into the log-->
        <property name="metricsLogFrequency" value="#{60 * 10 * 1000}"/>

    </bean>

    <!--
        Initialize property configurer so we can reference environment variables.
    -->
    <bean id="propertyConfigurer"
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
        <property name="systemPropertiesModeName"
value="SYSTEM_PROPERTIES_MODE_FALLBACK"/>
        <property name="searchSystemEnvironment" value="true"/>
    </bean>

    <!--
        Defines expiry policy based on moment of creation for ignite cache.
    -->
    <bean id="7_minutes_duration" class="javax.cache.expiry.CreatedExpiryPolicy"
factory-method="factoryOf" scope="prototype">
        <constructor-arg>
            <bean class="javax.cache.expiry.Duration">
                <constructor-arg value="MINUTES"/>
                <constructor-arg value="7"/>
            </bean>
        </constructor-arg>
    </bean>

    <!--
    Defines expiry policy based on moment of creation for ignite cache.
-->
    <bean id="3_hours_duration" class="javax.cache.expiry.CreatedExpiryPolicy"
factory-method="factoryOf" scope="prototype">
        <constructor-arg>
            <bean class="javax.cache.expiry.Duration">
                <constructor-arg value="HOURS"/>
                <constructor-arg value="3"/>
            </bean>
        </constructor-arg>
    </bean>

    <!--
    Defines expiry policy based on moment of creation for ignite cache.
-->
    <bean id="24_hours_duration" class="javax.cache.expiry.CreatedExpiryPolicy"
factory-method="factoryOf" scope="prototype">
        <constructor-arg>
            <bean class="javax.cache.expiry.Duration">
                <constructor-arg value="HOURS"/>
                <constructor-arg value="24"/>
            </bean>
        </constructor-arg>
    </bean>
</beans>
```

C.1 Enable Cache Expiry Policy

One feature to add to each one of the caches is the duration adding the expiryPolicyFactory property cache by cache.

```
<!-- Partitioned cache example configuration (Atomic mode). -->
<bean class="org.apache.ignite.configuration.CacheConfiguration">
…
<property name="expiryPolicyFactory">
 <bean class="javax.cache.expiry.CreatedExpiryPolicy" factory-method="factoryOf">
  <constructor-arg>
   <bean class="javax.cache.expiry.Duration">
    <constructor-arg value="MINUTES"/>
    <constructor-arg value="7"/>
   </bean>
  </constructor-arg>
 </bean>
</property>

</bean>
```

This is now default as shown by the bean "7_minutes_duration" above.

C.2 Enable TLSv1.2 Ignite nodes

In the configuration files, the TLSv1.2 is enabled for Ignite Nodes if at ignite.xml, the following is present:

```
<!-- Ssl/Tls context. -->
        <!--IMPORTANT: THIS IS JUST A DEMO CONFIGURATION AND DEMO KEYSTORES,
NEEDS TO BE CHANGED FOR PRODUCTION ALSO THE KEYS IN THE KEYSTORES NEED TO BE
CREATED AS WELL-->
        <property name="sslContextFactory">
            <bean class="org.apache.ignite.ssl.SslContextFactory">
                <property name="keyStoreFilePath"
value="${EIDAS_CONFIG_REPOSITORY}/ignite/KeyStore/server.jks"/>
                <property name="keyStorePassword" value="123456"/>
                <property name="trustStoreFilePath"
value="${EIDAS_CONFIG_REPOSITORY}/ignite/KeyStore/trust.jks"/>
                <property name="trustStorePassword" value="123456"/>
                <property name="protocol" value="TLSv1.2"/>
            </bean>
        </property>
```

Of course the corresponding keystores:

Will need to be present at

EIDAS-Config/ignite/KeyStore/server.jks

EIDAS-Config/ignite/KeyStore/trust.jks

This default configuration works with an Oracle (Sun) Java Runtime. In the case you are using IBM Java Runtime, you will need the additional "keyAlgorithm" property with the value "IBMX509" in the SslContextFactory configuration.

```
    <property name="sslContextFactory">
```

```
        <bean class="org.apache.ignite.ssl.SslContextFactory">
            <property name="keyAlgorithm" value="IBMX509" />
            …
            <property name="protocol" value="TLSv1.2" />
        </bean>
    </property>
```

For further information regarding this topic please refer to Ignite documentation pages, e.g. https://apacheignite.readme.io/docs/ssltls

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 *7.5 — 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 `hazelcastNode.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 caches are activated by the inclusion of the dependency eidas-jcache-hazelcast-node.jar.

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; or
- 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 15: 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 16 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 16: 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 17: 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 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.