eIDAS-Node Demo Tools Installation and Configuration Guide v2.6

eIDAS eID Implementation

Exported on 04/15/2022

Table of Contents

1 Introduction 5
1.1 Purpose 5
1.2 Document structure 5
1.3 Other technical reference documentation 5
2 Demo Products overview 7
2.1 Integration package 7
2.2 Modules 7
3 Setup configuration directories 10
4 Setting up the Demo Service Provider 11
5 Setting up the Demo Identity Provider 12
6 Setting up the Demo MS-Specific Connector 13
7 Setting up the Demo MS-Specific Proxy Service 14
8 Additional attributes 16
9 Preparing the installation 17
10 Building and deploying the software 18
10.1 Tomcat v9.0.x server deployment 18
10.2 WildFly 23.0.2 Final Server (Servlet-Only Distribution) deployment 20
10.3 WebLogic 14.1.1.0.0 Server deployment 21
10.4 WebSphere Liberty Server v21.0.0.5 (WebProfile 8) deployment 22
10.5 Monolithic Deployment 24
11 Verifying the installation 25
11.1 Tomcat 9 25
11.2 Wildfly 23.0.2 Final Server (Servlet-only Distribution) 25
11.3 Weblogic 14.1.1.0.0 25
11.4 Websphere Liberty 21.0.0.5 (Web Profile 8) 26
11.5 Configuration files 26
12 Simple protocol 27
12.1 Original SAML EIDAS Request information items 27

12.2 SimpleRequest example 27
12.3 Original SAML EIDAS Response information items 30
12.4 SimpleResponse example 31
13 Demo Tools Migration 36
13.1 Code changes 36
13.2 Configuration changes 36

Document history

Version	Date	Modification reason	Modified by
1.0	06/10/2017	Origination	DIGIT
2.0	11/04/2018	Rewritten for version 2.0 to take account of architectural changes with Demo Specific Connector and Demo Specific Proxy Service as well as Demo-SP, Demo IdP.	DIGIT
2.1	09/07/2018	Reuse of document policy updated and version changed to match the corresponding Release. Minor document clarifications made.	DIGIT
2.2	14/09/2018	Minor document clarifications made.	DIGIT
2.3	20/06/2019	Updates related to Jcache Ignite implementation. Other topics that are necessary to migrate.	DIGIT
2.4	06/12/2019	Minor document clarifications made.	DIGIT
2.5	11/12/2020	eIDAS-Node 2.5 release	DIGIT
2.6	04/2022	eIDAS-Node 2.6 release	DIGIT

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

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

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 installation and configuration settings for the Demo Tools (SP and IdP) supplied with the package for basic testing.

1.1 Purpose

The purpose of this document is to describe how to quickly install the Demo tools provided in the Integration Package (Service Provider (SP), Identity Provider (IdP), Specific Connector and Specific Proxy Service) for testing purposes.

Please note that this is not a guide for your national infrastructure, for implementation options please read the *eIDAS-Node National IdP and SP Integration Guide*.

1.2 Document structure

This document is divided into the following sections:

- Chapter 1 − *Introduction*: this section.
- Chapter 2 − *Demo Products overvie*w provides information on the deliverable including the package, the modules and dependencies.
- Chapter 3 — *Setup configuration directories* describes the setup configuration directories and environment variables.
- Chapter 4 — *Setting up the Demo Service Provider* provides information on the Demo SP properties to enable set up.
- Chapter 5 — *Setting up the Demo Identity Provider* provides information on the Demo IdP properties to enable set up.
- Chapter 6 — *Setting up the Demo MS-Specific Connector* provides information on the Demo MS-Specific Connector properties to enable set up.
- Chapter 7 — *Setting up the Demo MS-Specific Proxy Service* provides information on the Demo MS-Specific Proxy Service properties to enable set up.
- Chapter 8 —*Additional attributes* describes how to add attributes.
- Chapter 9 — *Preparing the installation* for this information you should refer to the eIDAS-Node Installation and Configuration Guide.
- Chapter 10 — *Building and deploying the software* describes the steps to build and then to deploy the software on the supported servers.
- Chapter 11 — *Verifying the installation* shows the final structure of your application server relevant directories.
- Chapter 12 — *Simple protocol* describes the implementation of Simple Protocol for communication between SP and Specific Connector, and Specific Proxy Service and IdP
- Chapter 13 — *Demo Tools Migration* provides a resume of the topics to be aware in the migration to 2.5 version from previous 2.6.

1.3 Other technical reference documentation

We recommend that you also familiarise yourself with the following eID technical reference documents which are available on **Digital Home > eID**

- *eIDAS-Node Installation, Configuration and Integration Quick Start Guide* describes how to quickly install a Service Provider, eIDAS-Node Connector, eIDAS-Node Proxy Service and 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 Installation and Configuration Guide* describes the steps involved when implementing a Basic Setup and goes on to provide detailed information required for customisation and deployment.
- *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 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.

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 Demo Products overview

This section provides information on the deliverable including the integration package, the modules and dependencies.

2.1 Integration package

The demo products deliverable consists of the following files:

- SP.war
- IdP.war
- SpecificConnector.war
- SpecificProxyService.war

These are web applications that can be deployed in most available Java web containers.

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.
Simple Protocol	EIDAS-SimpleProtocol	Simple Protocol implementation to demonstrate a MS-Specific protocol between SP and Specific-Connector and between IdP and Specific Proxy Service. **Not to be used in production**.
Commons	EIDAS-Commons	Common Applications components and utility classes for implementing functionality of authentication service.
JCache-Dev	EIDAS-JCache-Dev	Common code for Guava non-distributed JCache implementations

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-Specific-Communication	EIDAS-JCache-Ignite-Specific-Communication	Implementation of Ignite JCache 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.
MS Specific Protocol	EIDAS-SimpleProtocol	Module that provides the code to create simple protocol request and response used between the SP and Specific Connector and between IdP and Specific Proxy. Please see appendix for further details. **Not to be used in production**
MS Specific Connector	EIDAS-SpecificConnector	Demo implementation of Member State (MS) specific connector module. **Not to be used in production**.
MS Specific Proxy Service	EIDAS-SpecificProxyService	Demo implementation of Member State (MS) specific Proxy Service module. **Not to be used in production**
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**
Service provider	EIDAS-SP	Demo implementation of Service Provider module. **Not to be used in production**
Identity provider	EIDAS-IdP-1.0	Sample of Identity Provider module. **Not to be used in production**
Basic Setup configuration	EIDAS-Config	Sample configuration as in 12.2.

The figure below shows the dependencies between the installed modules. Note that the modules shown in red are labelled 'DO NOT USE' in the legend, this means use only as samples for demonstration purposes to show that the eIDAS-Node is working, do not use in production. Furthermore, several security vulnerabilities exist and deploying 'as is' in production carries significant risks.

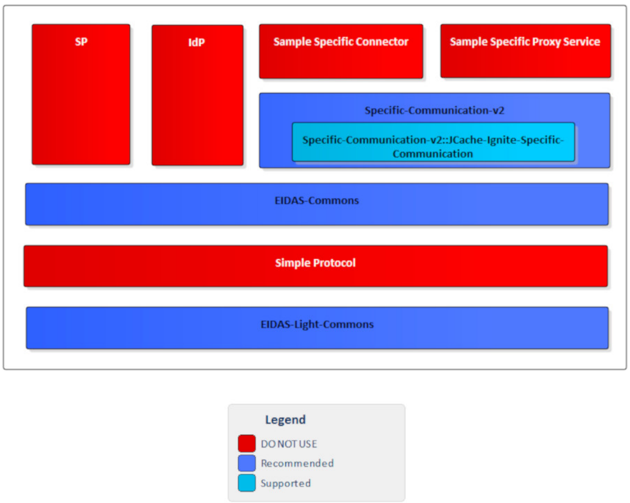
SP
IdP
Sample Specific Connector
Sample Specific Proxy Service
Specific-Communication-v2
Specific-Communication-v2::JCache-Ignite-Specific-Communication
EIDAS-Commons
Simple Protocol
EIDAS-Light-Commons
Legend
🟥 DO NOT USE
🟦 Recommended
🟦 Supported

Figure 1: Dependencies between the installed modules

3 Setup configuration directories

This section describes the setup configuration directories and environment variables.

There are five different environment variables used to locate the Demo Tools (Demo-SP, Demo IdP, Demo Specific Connector and Demo Specific Proxy Service) directories of configuration files. These can be defined as OS environment variables or setting it to the runtime environment (by –D switch to JVM or on the AS admin console).

Table 2: Setup configuration directories

Environment variable	Used in	Example target configuration directory
$SP_CONFIG_REPOSITORY	spApplicationContext.xml	file:/C:/PGM/projects/ configEidas/sp/
$SPECIFIC_CONNECTOR_CONFIG_REPOSITORY	specificConnectorApplicationContext.xml	file:/C:/PGM/projects/ configEidas/specificConnector/
$SPECIFIC_PROXY_SERVICE_CONFIG_REPOSITORY	specificProxyServiceEnvironmentContext.xml	file:/C:/PGM/projects/ configEidas/specificProxyService/
$IDP_CONFIG_REPOSITORY	idpApplicationContext.xml	file:/C:/PGM/projects/ configEidas/idp/
$EIDAS_CONFIG_REPOSITORY This configuration is used by the eIDAS-Node, please see *eIDAS-Node Installation and Configuration Guide*.	specificConnectorApplicationContext.xml specificProxyServiceApplicationContext.xml	file:/C:/PGM/projects/configEidas

By default OS environment variables 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 the following files:

- spEnvironmentContext.xml
- specificConnectorEnvironmentContext.xml
- specificProxyServiceEnvironmentContext.xml
- idpEnvironmentContext.xml

Please look inside these files to see how it is done.

4 Setting up the Demo Service Provider

This section provides information on the Demo SP properties to enable set up.

The Demo Service Provider (SP) can be used to simulate an MS SP requesting authentication. It works with the default MS-Specific-Connector part using the simple protocol language.

The Basic Setup provides a preconfigured version of Demo Service Provider, however you may need to fine-tune some options.

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

Table 3: Service Provider Properties

Key	Description
provider.name	Provider Name for this Service Provider
sp.return	URL used when the eIDAS-Node Connector finishes the process. This must be the value of the machine running the Service Provider, its format is *http://sp.ip.address:sp.port.number/sp.deployment.name/ReturnPage.*

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

Table 4: Available eIDAS-Node for Service Provider

Key	Description
country.number	The number of possible eIDAS-Nodes that can communicate with this SP
countryX.name	The name of the eIDAS-Node X(= positive integer)
countryX.url	The URL for the eIDAS-Node X. This must be the value of the machine running the eIDAS-Node using the format: *http://node.ip.address:node.port.number/node.deployment.name/* This URL is used by the SP to send its request.

5 Setting up the Demo Identity Provider

This section provides information on the Demo IdP properties to enable set up.

The Demo Identity Provider (IdP) can be used to simulate an MS IdP requesting authentication. It works with the default MS-Specific-Proxy-Service part using the simple protocol language.

In order to proceed with the Basic Setup, you may need to modify the configuration of the Demo Identity Provider.

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

The *idp.properties* is used by the IdP to provide the attribute values in the format: *<username>.<attributeName>=<attributeValue>*.

Table 5: Sample of user.properties content

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

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

Table 6: Identity Provider Properties

Key	Description
idp.issuer	Issuer name for the IdP.

6 Setting up the Demo MS-Specific Connector

This section provides information on the Demo MS-Specific Connector properties to enable set up.

The eIDAS-Node integration package contains a Demo Member State Specific Connector part that is aligned with the use of Demo SP.

There are some configuration items that might need to be customised according to the test environment.The configuration file name is *specificConnector.xml,* and is located by SPECIFIC_CONNECTOR_CONFIG_REPOSITORY environment variable or command line argument.

Table 7: Specific Connector part properties

Key	Description
issuer.name	Name of the issuer. Responses sent will have this value as issuer.
distributedMapsSpecificConnector	Boolean value (true | false), which indicates if the application will activate distributed maps feature, necessary if clusters are used.
specific.connector.request.url	The URL of the Node to send the binary light token related to the Light Request.
relaystate.randomize.null	Boolean value (true | false), to activate or de-activate the behaviour of populating a null relayState with a random value.

7 Setting up the Demo MS-Specific Proxy Service

This section provides information on the Demo MS-Specific Proxy Service properties to enable set up.

The eIDAS-Node integration package contains a Demo Member State Specific Proxy Service part that is aligned with the use of Demo IdP.

There are some configuration items that might need to be customised according to the test environment. The configuration file name is *specificProxyService.xml*, and is located by SPECIFIC_PROXY_SERVICE_CONFIG_REPOSITORY environment variable or command line argument.

Table 8: Specific part properties

Key	Description
issuer.name	Name of the issuer for the IdP. Responses sent will have this value as issuer.
distributedMapsSpecificProxyService	Boolean value (true | false), which indicates if the application will activate distributed maps feature, to be used in cluster mode.
idp.url	URL to where the MS request will be sent.
specific.proxyservice.idp.response.service.url	URL to where the MS Specific Proxy Service can receive the response from the Demo IdP. It is send in the request to the IdP.
ask.consent.request	Boolean value (true | false), which indicates if the application will activate the consent pages for the request. If set to "true", the Consent Page will be displayed to the user when processing the request from the eIDAS-Node Connector. Attributes without consent will be removed from the response.
ask.consent.response	Boolean value (true | false), which indicates if the application will activate the consent pages for the response. If set to "true", the Value Consent Page (CV) will be displayed before sending the response to the eIDAS-Node Connector. The user is able to cancel the forwarding of authentication data, resulting in an authentication failure.
ask.consent.response.show.only.eidas.attributes	Boolean value (true | false), which indicates if the application will activate the display of the response's attribute names. Depends on activation of ask.consent.response If set to "true" only the Core eIDAS attributes/values will be displayed. On "false", the Value Consent Page (CV) will display all the Response attributes/values, including additional (specified in XML file) ones.

Key	Description
ask.consent.response.show.attribute.values	Boolean value (true | false), which indicates if the application will activate the display of the response's attribute values. Depends on activation of ask.consent.response If set to "true", the Value Consent Page (CV) will display attribute names and values for the Response, "false" will result in attribute names only.
consent.Request.LightToken.Secret	Secret to be used in the request consent.
consent.Request.LightToken.Algorithm	Digest Algorithm for the request consent
consent.Response.LightToken.Secret	Secret to be used in the response consent
consent.Request.LightToken.Algorithm	Digest Algorithm for the response consent
default.specific.proxyservice.idp.response.service.url	URL where the MS Specific Proxy Service can receive the response from the Demo IdP. It is sent in the request to the IdP when specific modules are included in the Node as JAR.
specific.proxyservice.response.url	The URL of the Node to send the binary light token related to the Light Response.
relaystate.randomize.null	Boolean value (true | false), to activate or de-activate the behaviour of populating a null relayState with a random value.

8 Additional attributes

This section describes how to add attributes.

To add additional attributes use the files named *additional-attributes.xml*, located in the environment variables:

- $SPECIFIC_CONNECTOR_CONFIG_REPOSITORY
- $SPECIFIC_PROXY_SERVICE_CONFIG_REPOSITORY

or by command line argument. The file *eidas-attributes.xml* should remain unchanged.

The following table contains the additional attribute keys that need to be present to add an additional attribute.

Table 9: Additional attributes

Key	Description
1.NameUri	URI of the attribute.
1.FriendlyName	Friendly name of the attribute.
1.PersonType	PersonType, either natural or legal , corresponding to the Natural and Legal Persons
1.Required	If the attribute is to be set as required.
1.XmlType.NamespaceUri	The additional attribute namespace URI.
1.XmlType.LocalPart	The additional attribute local part.
1.XmlType.NamespacePrefix	The additional attribute's namespace prefix.
1.AttributeValueMarshaller	The additional attribute's namespace value marshaller.

To add a second attribute you will need to increment the prefix number (i.e. the additional attribute would be prefixed "2" and so on).

Also the same has to be done in the eIDAS-Node configuration file for these additional attributes to be recognised.

9 Preparing the installation

For instructions on how to prepare the servers: Tomcat, WildFly, WebLogic or WebSphere before deploying the Demo Tools please refer to the *eIDAS-Node Installation and Configuration Guide*.

10 Building and deploying the software

This section describes the steps to build and then to deploy the software on the supported servers.

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 3.3.9 and above. Lower versions can result in exceptions.

There are two ways to build the binaries from sources:

- **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: we need only one profile just for weblogic application server named weblogic
  Second: two profiles related to the scope of modules to be build, 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.

- **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 don't forget 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.

10.1 Tomcat v9.0.x server deployment

You must compile, install and deploy the projects, either by compiling the parent project (Table 10: Parent project build for Tomcat Server) or by compiling each module separately in the order shown below (Table 11: Module-based build for Tomcat Server).

At a command prompt, first navigate to the folder indicated and then enter the corresponding command line.

In order to deploy the project, after the build is complete, copy the artifacts needed IdP.war, SP.war, SpecificConnector.war and SpecificProxyService.war to the deploy folder of the Server. The EidasNode.war may also be needed, check eIDAS-Node Installation and configuration guide for more information.

Deploy folder for tomcat is: $TOMCAT_HOME/webapps

Table 10: Parent project build for Tomcat Server

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install –P [NodeOnly],DemoToolsOnly [-P specificCommunicationJcacheIgnite] [-DspecificJar]

Table 11: Module-based build for Tomcat Server

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-Specific-Communication	mvn clean install
6	EIDAS-JCache-Ignite	mvn clean install
7	EIDAS-JCache-Ignite-Specific-Communication	mvn clean install
8	EIDAS-SpecificCommunication Definition	mvn clean install [-P specificCommunicationJcacheIgnite, specificCommunicationJcacheDev] [-DspecificJar]
9	EIDAS-Updater	mvn clean install
10	SimpleProtocol	mvn clean install
11	EIDAS-SpecificConnector	mvn clean install
12	EIDAS-SpecificProxyService	mvn clean install
13	EIDAS-SP	mvn clean package
14	EIDAS-IdP-1.0	mvn clean package

10.2 WildFly 23.0.2 Final Server (Servlet-Only Distribution) deployment

You must compile, install and deploy the projects, either by compiling the parent project (Table 14: Parent project build for WildFly 23.0.2 Final Server (Servlet-Only Distribution)) or by compiling each module separately in the order shown below (Table 15: Module-based build for WildFly 23.0.2 Final Server (Servlet-Only Distribution)).

At a command prompt, first navigate to the folder indicated and then enter the corresponding command line.

In order to deploy the project, after the build is complete, copy the artifacts needed IdP.war, SP.war, SpecificConnector.war and SpecificProxyService.war to the deploy folder of the Server. The EidasNode.war may also be needed, check eIDAS-Node Installation and configuration guide for more information.

Deploy folder for wildfly is: $WILDFLY_HOME/standalone/deployments

Table 14: Parent project build for WildFly 23.0.2 Final Server (Servlet-Only Distribution)

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install –P [NodeOnly],DemoToolsOnly,wildfly [-P specificCommunicationJcacheIgnite] [-DspecificJar]

Table 15: Module-based build for WildFly 23.0.2 Final Server (Servlet-Only Distribution)

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-Specific-Communication	mvn clean install
6	EIDAS-JCache-Ignite	mvn clean install
7	EIDAS-JCache-Ignite-Specific-Communication	mvn clean install
8	EIDAS-SpecificCommunicationDefinition	mvn clean install [-P specificCommunicationJcacheIgnite] [-DspecificJar]

Step	Folder	Command line
9	EIDAS-Updater	mvn clean install
10	SimpleProtocol	mvn clean install
11	EIDAS-SpecificConnector	mvn clean install
12	EIDAS-SpecificProxyService	mvn clean install
13	EIDAS-SP	mvn clean package -Pwildfly
14	EIDAS-IdP-1.0	mvn clean package -Pwildfly

10.3 WebLogic 14.1.1.0.0 Server deployment

You must compile, install and deploy the projects, either by compiling the parent project (Table 16: Parent project build for WebLogic Server) or by compiling each module separately in the order shown below (Table 17: Module-based build for WebLogic Server).

At a command prompt, first navigate to the folder indicated and then enter the corresponding command line.

In order to deploy the project, after the build is complete, copy the artifacts needed IdP.war, SP.war, SpecificConnector.war and SpecificProxyService.war to the deploy folder of the Server. The EidasNode.war may also be needed, check eIDAS-Node Installation and configuration guide for more information.

Deploy folder for weblogic is: $MW_HOME/user_projects/domains/base_domain/autodeploy/

Table 16: Parent project build for WebLogic Server

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install –P weblogic[,NodeOnly],DemoToolsOnly

Table 17: Module-based build for WebLogic Server

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install
2	EIDAS-Light-Commons	mvn clean install

Step	Folder	Command line
3	EIDAS-Commons	mvn clean install
4	EIDAS-JCache-Dev	mvn clean install
5	EIDAS-JCache-Dev-Specific-Communication	mvn clean install
6	EIDAS-JCache-Ignite	mvn clean install
7	EIDAS-JCache-Ignite-Specific-Communication	mvn clean install
8	EIDAS-SpecificCommunicationDefinition	mvn clean install [-P specificCommunicationJcacheIgnite] [-DspecificJar]
9	EIDAS-Updater	mvn clean install
10	SimpleProtocol	mvn clean install
11	EIDAS-SpecificConnector	mvn clean install –P weblogic
12	EIDAS-SpecificProxyService	mvn clean install –P weblogic
13	EIDAS-SP	mvn clean package –P weblogic
14	EIDAS-IdP-1.0	mvn clean package –P weblogic

10.4 WebSphere Liberty Server v21.0.0.5 (WebProfile 8) deployment

You must compile, install and deploy the projects, either by compiling the parent project (Table 18: Parent project build for WebSphere Server) or by compiling each module separately in the order shown below (Table 19: Module-based build for WebSphere Server).

At a command prompt, first navigate to the folder indicated and then enter the corresponding command line.

In order to deploy the project, after the build is complete. Copy the artifacts needed (IdP.war, SP.war, SpecificConnector.war and SpecificProxyService.war) to the deploy folder of the server.

The EidasNode.war may also be needed, check eIDAS-Node Installation and configuration guide for more information.

Deploy folder for Websphere Liberty (WebProfile) is: ${WEBSPHERE_SERVER_HOME}/usr/servers/${SERVER}/dropins/

Table 18: Parent project build for WebSphere Server

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install –P [NodeOnly],DemoToolsOnly [-P specificCommunicationJcacheIgnite] [-DspecificJar]

Table 19: Module-based build for WebSphere Server

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-Specific-Communication	mvn clean install
6	EIDAS-JCache-Ignite	mvn clean install
7	EIDAS-JCache-Ignite-Specific-Communication	mvn clean install
8	EIDAS-SpecificCommunication Definition	mvn clean install [-P specificCommunicationJcacheIgnite] [-DspecificJar]
9	EIDAS-Updater	mvn clean install
10	SimpleProtocol	mvn clean install
11	EIDAS-SpecificConnector	mvn clean install
12	EIDAS-SpecificProxyService	mvn clean install

Step	Folder	Command line
13	EIDAS-SP	mvn clean package
14	EIDAS-IdP-1.0	mvn clean package

10.5 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-SpecificConnector
- EIDAS-SpecificProxyService modules
- EIDAS-SP
- EIDAS-IdP-1.0

This also applies to EidasNode modules, so please check the *Monolithic Deployment* section in the *eIDAS-Node Installation and Configuration Guide* for more details.

11 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.

11.1 Tomcat 9

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

11.2 Wildfly 23.0.2 Final Server (Servlet-only Distribution)

```
$WILFDLY_HOME /standalone/Deployments/
EidasNode.war
IdP.war
SpecificConnector.war
SpecificProxyService.war
SP.war
(server specific directories were not included)
```

11.3 Weblogic 14.1.1.0.0

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

11.4 Websphere Liberty 21.0.0.5 (Web Profile 8)

```
$WEBSPHERE_SERVER_HOME /usr/servers/ $SERVER /dropins/
EidasNode.war
IdP.war
SpecificConnector.war
SpecificProxyService.war
SP.war
(server specific directories were not included)
```

11.5 Configuration files

The following configuration and keystore files are needed for the full installation with Demo Tools. The layout itself can be different, depending on the environment variables, so this is just an example of Basic Setup:

- server/igniteSpecificCommunication.xml (needed only if profile specificCommunicationJcacheIgnite is activated)
- server/idp/additional-attributes.xml
- server/idp/eidas-attributes.xml
- server/idp/idp.properties
- server/idp/user.properties
- server/sp/additional-attributes.xml
- server/sp/eidas-attributes.xml
- server/sp/sp.properties
- server/specificConnector/additional-attributes.xml
- server/specificConnector/eidas-attributes.xml
- server/specificConnector/specificCommunicationDefinitionConnector.xml
- server/specificConnector/specificConnector.xml
- server/specificProxyService/additional-attributes.xml
- server/specificProxyService/eidas-attributes.xml
- server/specificProxyService/specificCommunicationDefinitionProxyservice.xml
- server/specificProxyService/specificProxyService.xml

12 Simple protocol

Simple Protocol has been implemented for communication between SP and Specific Connector, and Specific Proxy Service and IdP. The main goal is to show the concept of integrating SPs, IdPs or similar entities with an eIDAS-Node. This is a simplified protocol for demonstration purposes only. It does not include security features.

The Simple Protocol was not designed to be used 'as is' by Member States, only for demonstration purposes. Some parts of it may evolve/be changed in future versions.

12.1 Original SAML EIDAS Request information items

Request

```
AuthnRequest
    ID
    Destination
    ForceAuthn
    IssueInstant
    ProviderName
    Version
    AssertionConsumerServiceURL
    SPType
    RequestedAuthnContext
        Comparison
        AuthnContextClassRef
    RequestedAttributes
        RequestedAttribute
            FriendlyName
            isRequired
            Value
                LatinScript
            Value
```

12.2 SimpleRequest example

SimpleRequest

```
{
    "authentication_request": {
        "attribute_list": [
            {
                "type": "requested_attribute",
                "name": "D-2012-17-EUIdentifier",
                "required": false
            },
            {
                "type": "requested_attribute",
                "name": "EORI",
                "required": false
            },
            {
                "type": "requested_attribute",
                "name": "LEI",
                "required": false
            },
            {
                "type": "requested_attribute",
                "name": "LegalName",
                "required": true
            }
        ],
        "requested_authentication_context": {
            "comparison": "minimum",
            "context_class": [
                "A"
            ],
            "non_notified_context_class": [
                " http://non.eidas.eu/NotNotified/LoA/1 "
            ]
        },
        "citizen_country": "CA",
        "created_on": "2020-06-12T18:11:25.107+02:00",
        "force_authentication": true,
        "id": "835a09f7-0179-4198-9c19-36680d0fe531",
        "name_id_policy": "unspecified",
        "provider_name": "DEMO-SP-CA",
        "requester_id": " http://eidas.eu/EidasNode/RequesterId ",
        "serviceUrl": " http://localhost:8080/SP/ReturnPage ",
        "sp_type": "public",
        "version": "1"
    }
}
```

Note: If an attribute value is supplied in the Request, that will be a valueattribute, so "type" will change from "requested_attribute" to a certain type.

<table>
<tr><th>Simple Protocol</th><th>LightRequest</th><th>Mandatory Yes/No</th><th>Nature</th></tr>
<tr><td>authentication_request</td><td>lightRequest</td><td>No</td><td>abstract</td></tr>
<tr><td>version</td><td></td><td>Yes</td><td>always "1"</td></tr>
<tr><td>id</td><td>id to map</td><td>Yes</td><td>UUID generated</td></tr>
<tr><td>created_on</td><td></td><td>Yes</td><td>timestamp, local tim</td></tr>
<tr><td>force_authentication</td><td></td><td>No</td><td>always "true"</td></tr>
<tr><td>provider_name</td><td>providerName</td><td>No</td><td>string</td></tr>
<tr><td>serviceUrl</td><td></td><td>Yes</td><td>URL for the Respons</td></tr>
<tr><td>sp_type</td><td>spType</td><td>No</td><td>"public" | "private" |</td></tr>
<tr><td>requester_id</td><td>requesterId</td><td>No</td><td>string</td></tr>
<tr><td>context_class</td><td>levelOfAssurance type=”notified”</td><td rowspan="2">Yes</td><td rowspan="2"><table><tr><th>Context_class</th></tr><tr><td>"A" | "B"</td></tr><tr><td>"C" | "D"</td></tr><tr><td>"E"</td></tr></table>+ Optionally non not</td></tr>
<tr><td>non_notified_context_class</td><td>levelOfAssurance type=”nonNotified”</td></tr>
<tr><td>citizen_country</td><td>citizenCountryCode</td><td>Yes</td><td>This was an HTTP pa
Value: ISO Country C</td></tr>
<tr><td>name_id_policy</td><td>nameIDPolicy</td><td>No</td><td>Can be omitted OR a
To map:
persistent => urn:oa
transient => urn:oas
unspecified =&gt; urn:o</td></tr>
</table>

Simple Protocol	LightRequest	Mandatory Yes/No	Nature
attribute_list	immutableAttributeMap (please check example above)	No	Abstract, the idea is the AttributeRegistr add a prefix such as

Attribute type is always 'requested_attribute' for Request.

12.3 Original SAML EIDAS Response information items

Response

```
Response
        Destination
        ID
        InResponseTo
        IssueInstant
        Version
        Issuer
        Status
                StatusCode
                        StatusCode
                StatusMessage
        Assertion
                Issuer
                Subject
                        NameID
                                NameQualifier
                                Value
                        SubjectConfirmation
                                Method
                                SubjectConfirmationData
                                        Address
                                        InResponseTo
                                        NotOnOrAfter
                                        Recipient
                Conditions
                        NotBefore
                        NotOnOrAfter
        AudienceRestriction
                Audience
        AuthnStatement
                AuthnInstant
```

```
                        AuthnContext
                                AuthnContextClassRef
                                AuthnContextDecl
        AttributeStatement
                Attribute
                        FriendlyName
                        Name
                        NameFormat
                        AttributeValue
                                LatinScript
```

12.4 SimpleResponse example

SimpleResponse

Success:

```
{
    "response": {
        "version": "1",
        "id": "0a88c46e-24a7-4194-90f1-35485977bb18",
        "destination" : "http://", <-------------- TO BE DECOMISSIONED, NO
EIDINT yet
        "inresponse_to": "e7d5db08-0818-449f-bec2-d257bf9593d7",
        "created_on": "2012-04-23T20:28:43.511+02:00",
        "authentication_context_class": "high",
        "client_ip_address": "123.0.0.2",
        "issuer": "DEMO-IDP",
        "subject": "ES/BE/0123456",
        "name_id_format": "transient",
        "status": {
            "status_code": "success",
        },
        "attribute_list": [
            {
                "type": "string",
                "name": "gender",
                "value": "Male"
            },
            {
                "type": "string_list",
                "name": "birth_name",
                "values": [
                    {
                        "latin_script": false,
                        "value": "Árvíztűrő Tükörfúrógép"
                    },
                    {
                        "value": "Arvizturo Tukorfurogep"
                    }
                ]
            },
            {
                "type": "date",
                "name": "date_of_birth",
                "value": "1905-04-20"
            },
            {
                "type": "address",
                "name": "current_address",
                "value": {
                    "po_box": "1234",
                    "locator_designator": "28",
                    "locator_name": "DIGIT building",
                    "cv_address_area": "Etterbeek",
                    "thoroughfare": "Rue Belliard",
                    "post_name": "ETTERBEEK CHASSE",
                    "admin_unit_first_line": "BE",
                    "admin_unit_second_line": "ETTERBEEK",
                    "post_code": "1040"
                }
            }
        ]
    }
}
```

Error:

```
{
   "response" : {
      "version" : "1",
      "id" : "0a88c46e-24a7-4194-90f1-35485977bb18",
      "inresponse_to" : "e7d5db08-0818-449f-bec2-d257bf9593d7",
      "created_on" : "2012-04-23T20:28:43.511+02:00",
      "issuer" : "DEMO-IDP",
      "status" : {
         "status_code" : "failure",
         "sub_status_code" : "AuthnFailed",
         "status_message" : "all hands on deck"
      }
}
```

Simple Protocol	LightResponse	Mandatory Yes/ No	Nature
response	LightResponse	No	abstract
version		Yes	always "1"
id	ID to map	Yes	UUID generated
inresponse_to	Original req ID to map	Yes	Mandatory
subject	Subject	No	New field for the *user.properties* (eg.: xavi.subject) Only if message is SUCCESS!
name_id_format	NameIdFormat	No	At the IDP, copy the value of NameIDPolicy from the Request Only if message is SUCCESS!
client_ip_address	IPAddress	No	optional address of the client browser
created_on		Yes	timestamp, local time in json "de facto" format
authentication_context_class	LevelOfAssurance	No	"high" | "substantial" | "low" Or non-notified LoA values
issuer	Issuer	Yes	string
status	Status	Yes	abstract structure

Simple Protocol	LightResponse	Mandatory Yes/No	Nature
status_code	StatusCode	No	mandatory, allowed values: success | failure To be mapped as full SAML2Core URN (see SAML2Core): success => "*urn:oasis:names:tc:SAML:2.0:status:Success*" responder failure => "*urn:oasis:names:tc:SAML:2.0:status:Responder*" responder failure => "*urn:oasis:names:tc:SAML:2.0:status:Requester*" (not covered: "*urn:oasis:names:tc:SAML:2.0:status:VersionMismatch*" because it is for the Proxy Node in our simple implementation)
sub_status_code	SubStatusCode	No	To be mapped as SAML:Core secondary status code like AuthnFailed, attach this string to the URN (see SAML2Core), optional: only in case of failure. Possible values: AuthnFailed | InvalidAttrNameOrValue | InvalidNameIDPolicy | NoAuthnContext | NoAvailableIDP | NoPassive | NoSupportedIDP | PartialLogout | PartialLogout | PartialLogout | PartialLogout | ProxyCountExceeded | RequestDenied | RequestUnsupported | RequestVersionDeprecated | RequestVersionTooHigh | RequestVersionTooLow | ResourceNotRecognized | TooManyResponses | UnknownAttrProfile | UnknownPrincipal | UnsupportedBinding The strategy here is just to append "*urn:oasis:names:tc:SAML:2.0:status:*" in Specific Proxy, and remove it in the Specific Connector. The IDP should implement some of these (as appropriate) but not all e.g.: AuthnFailed should be the failure case when the credentials entered in the IDP are wrong.
status_message	StatusMessage	No	Only in case of failure. IDP should be able to produce some example text (e.g. "failed to authenticate because of bad credentials" for the "AuthnFailed" code)

Simple Protocol	LightResponse	Mandatory Yes/ No	Nature
attribute_list	ImmutableAttributeMap (please check example above)	No	Abstract, the idea is to use the FriendlyName attribute of EIDAS attributes here, then the AttributeRegistry.getByFriendlyName can be used in the mapping. It is possible to add a prefix such as "idp_". Only if message is SUCCESS!

Possible attribute types are: *string*, *string_list*, *date* and *address*. Add JAXB implementing class if more required.

13 Demo Tools Migration

In this section it is briefly described, the relevant changes in the Demo Tools worth mentioning occurred from previous version related either to code or configuration.

13.1 Code changes

In the context of the migration to OpenSaml 4, which requires an upgrade to Java 11, some few changes were done so the Ignite configuration can work.

New "jboss-deployment-structure.xml" files were added in the SpecificConnector and SpecificProxyService module and the following dependency had to be added to their content.

```
<dependencies>
     <module name= "jdk.unsupported" />
</dependencies>
```

A new "copy-resources" plugin to copy the jboss-deployment-structure.xml files to the war archives, only activated with the Wildfly profile was added to both pom.xml files from the SpecificConnector and SpecificProxyService modules

To allow that static files in webapp/resources folder could be not loaded in Weblogic 14.1.1.0.0, the folder webapp/resources was renamed to webapp/resource in both demo SP and demo IdP modules.

13.2 Configuration changes

In the context of the migration to OpenSaml 4, which requires an upgrade to Java 11, the supported list of servers has been changed with server that can support java 11.

Therefore the supported list of servers is now the following :

- Tomcat v9.0.x
- Wildfly 23.0.2 Final (Servlet Distribution)
- Weblogic 14.1.1.0.0
- WebSphere Liberty 21.0.0.5 (WebProfile 8)

Relative to the add of support for Justice and Consumers Financial Stability, Financial Services and Capital Markets Union (AKA BORIS) sector specific attributes, the files additional-attributes.xml were updated so that the attributes can be supported by Demo components and user.properties was updated so that values can be sent by user xavi for those new attributes. The files package.properties and package_en.properties in Demo Specific Proxy-Service were updated so that these sector specific attributes have a correct labels in the consent pages.