eIDAS-Node Migration Guide

For Version 2.3

Document history

Version	Date	Modification reason	Modified by
1.0	20/06/2019	Information how to migrate to eIDAS-Node v2.3 from eIDAS-Node v2.2	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

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 purpose of this document is to facilitate migration to eIDAS-Node v2.3 from eIDAS-Node v2.2.

1.1. Document structure

This document is divided into the following sections:

Chapter 1 — *Introduction*: this section.

Chapter 2 — *Prerequisites*: Identifies any prerequisites that are required before migrating your eIDAS-Node to version 2.3.

Chapter 3 — Changes: Contains detailed information about the changes that should be taken into consideration when migrating to eIDAS-Node version 2.3.

1.2. Document aims

The main aim of this document is to provide information on all the changes requiring your action when migrating to eIDAS-Node version 2.3, including:

- configuration changes; and
- changes to code.

2. Prerequisites

Before starting your migration to eIDAS-Node version 2.3 you should have:

- already implemented eIDAS-Node version 2.2;
- downloaded the eIDAS-Node v2.3 Integration Package; and
- downloaded the latest documentation.

3. Changes

3.1. Summary of changes

In the eIDAS-Node version 2.3 there are several main changes that affect your installation. The main changes are:

JCache support for Distributed Caches (see section 3.3);

Improved message logging of incoming/outgoing requests/responses (see section 3.8)

- (EID-652) Problem in validation of entityID of SP
- (EID-814) Validate Signature flag not working for unmarshall response
- (EID-914) LightRequest and LightResponse exposed to DTD threats

3.1.1. Bouncycastle CVEs

- (EID-859) CVE-2016-1000338
- (EID-860) CVE-2016-1000339
- (EID-861) CVE-2016-1000340
- (EID-862) CVE-2016-1000341
- (EID-863) CVE-2016-1000342
- (EID-864) CVE-2016-1000343
- (EID-865) CVE-2016-1000344
- (EID-866) CVE-2016-1000345
- (EID-867) CVE-2016-1000346
- (EID-868) CVE-2016-1000352
- (EID-869) CVE-2017-13098
- (EID-870) CVE-2018-1000613
- (EID-912) CVE-2018-1000180

3.1.2. Bootstrap CVEs

- (EID-800) Vulnerable version of Bootsrap
- **(**EID-854) CVE-2018-14040
- (EID-856) CVE-2018-14042
- (EID-857) CVE-2019-8331
- **(**EID-864) CVE-2016-1000343

3.1.3. JQuery CVEs

- (EID-799) Vulnerable version of JQuery
- (EID-858) CVE-2015-9251
- (EID-909) CVE-2019-11358

3.1.4. Spring CVEs

- (EID-754) CVE-2014-3625
- (EID-755) CVE-2015-0201
- (EID-756) CVE-2015-3192
- (EID-757) CVE-2015-5211
- (EID-758) CVE-2016-5007
- (EID-759) CVE-2018-1270
- (EID-760) CVE-2018-1271
- (EID-761) CVE-2018-1272
- (EID-911) CVE-2018-1199

For other fixes and improvements requiring no action, please see section 3.19.

3.2. Upgrade of Spring version

The spring version used was upgraded from 4.1.0.RELEASE to 4.3.18.RELEASE due to several CVE's that impacted the previous version and due to introduction of Jcache support. See section 3.1.4 Spring CVEs

3.2.1. Code changes

Some minor Junit tests corrections were necessary: adapted the expected message in Junit test, change path separator character. Note, with this change it is also recommended to use Maven version 3.5.4.

3.3. JCache support for Distributed Caches

The Communication Caches of Specific Communication Definition used for MS to Node communication and the caches used by the Node in a distributed environment were refactored to use JCache (JSR-000107 JCACHE JavaTM Temporary Caching API). The default distributed cache implementation used for the Specific communication cache is now Ignite instead of Hazelcast, previously. This release also introduced the possibility to have different Jcache implementations for the Node and for the SpecificCommunicationCaches.

3.3.1. Code changes

First of all, java version used in the build is now 1.8 instead of 1.7, previously. It is also recommended to use Maven version 3.5.4.

This change incurred the creation of 9 new modules:

- EIDAS-JCache-Dev
- EIDAS-JCache-Dev-Node
- EIDAS-JCache-Dev-Specific-Communication

- EIDAS-JCache-Hazelcast
- EIDAS-JCache-Hazelcast-Node
- EIDAS-JCache-Hazelcast-Specific-Communication
- EIDAS-JCache-Ignite
- EIDAS-JCache-Ignite-Node
- EIDAS-JCache-Ignite-Specific-Communication

The modules hold the implementation of the several possibilities, for JCache implementation. EIDAS-JCache-Dev-X and EIDAS-JCache-Hazelcast-X hold the previous version possibilities: local maps using Guava or distributed maps using Hazelcast, respectively. The EIDAS-JCache-Ignite-X modules hold the code for the new Ignite implementation.

In Module EIDAS-Commons, new classes AbstractCache, CommunicationCache and ConcurrentCacheService were added. These replace the previous hierarchy to implement Cache. This implied a new dependency to be added javax.cache:cache-api:1.1.0

EIDAS-SpecificCommunicationDefinition module

In SpecificConnectorCommunicationServiceImpl and SpecificProxyserviceCommunicationServiceImpl classes the CommunicationCache class was replaced by another class with same name that implements javax.cache.Cache. This implied replacing the call of method remove from previous CommunicationCache by new one getAndRemove, defined at javax.cache.Cache interface.

In SpecificCommunicationDefinitionBeanNames there were changes regarding the names of the beans that refer to the caches, basically removing "MAP" and replacing it by "CACHE".

The ServletContextListenerImpl.java and CommunicationCache.java and config folder were deleted.

Some existing profiles were changed or new ones were added. The previous profile specificProxyServiceWarPackaging was renamed to specificCommunicationJcacheIgnite, activated by default. The previous profile specificProxyServiceJarPackaging was replaced by the specificCommunicationJcacheDev.

Two new profiles were added: specificCommunicationJcacheHazelcast and specificCommunicationJcacheProvidedImpl. The first one activated the use of Hazelcast for communication caches. The second one, when activated, does not include any JCache implementation. This one aims to allow adding JCache implementation after the build is finished, e.g., on the server directly.

So in conclusion, regarding the profiles for Jcache implementations, now it exists the following ones:

- Three profiles for the node:
  - nodeJcacheIgnite : Ignite JCache implementation;
  - nodeJcacheHazelcast: Hazelcast 3.2 JCache implementation/adaptation;
  - nodeJcacheDev : Guava JCache implementation/adaptation (Only for non-distributed case.)

- Three profiles for the specificCommunication:
    - specificCommunicationJcacheIgnite : Ignite JCache implementation;
    - specificCommunicationJcacheHazelcast: Hazelcast 3.2 JCache implementation/adaptation;
    - specificCommunicationJcacheDev: Local Caches implementation (Possible only for Monolithic deployment.)
- One profile for the specificCommunication, without any JCache implementation:
    - specificCommunicationJcacheProvidedImpl: to be provided for JCache implementation.

When building the node, one profile for the node and one for the specificCommunication must be chosen, please check eIDAS-Node Migration Guide for more details.

Changed the specificCommunicationDefinitionApplicationContext.xml, moving the beans that reference the caches implementation to the eidas-jcahe implementations, inside jCacheImplBeans.xml.

Relative to applicationContext.xml at EIDAS-Node module, several changes were introduced such as renaming of spring caches beans' ids used by the node that ended with Prod or Dev and thus now they end with Impl. Due to that the now unnecessary beans: runenvironment, useDistributedMaps and distributedEnvSuffix were removed from node's applicationContext.xml

However, now several combinations are possible. Thus, the logic used on the Specific Communication Caches should be used and therefore the ending should be "Impl".

Therefore, we suggest you do the necessary changes for that renaming.

> Remove the bean distributedEnvSuffix if it is not necessary any more at the applicationContext.xml.

The following Junit tests were also modified:

- SpecificConnectorCommunicationServiceImplTest,
- SpecificProxyserviceCommunicationServiceImplTest,
- SpecificCommunicationApplicationContextProviderTest

3.3.2. Configuration changes

Four new files were introduced and previous hazelcast.xml was removed:

- hazelcastNode.xml:
    - defines the "caches" used in the Node;
- hazelcastSpecificCommunication.xml:
    - defines the "caches" used in the specific communication definition;
- igniteSpecificCommunication.xml:
    - stores the configuration for Ignite's specific communication caches
- igniteNode.xml:

- stores the configuration for Ignite's Node caches.

Note that, if only Ignite is used, there is no need to have and to include hazelcastSpecificCommunication.xml in $EIDAS_CONFIG_FOLDER. Similarly, when only Hazelcast is chosen for communication caches, there is no need to have ignite.xml.

Relatively to hazelcast.xml, the communication caches that were defined were moved to the new hazelcastSpecificCommunication.xml file.

In $EIDAS_CONFIG_FOLDER/eidas.xml the distributedMaps entry was removed.

3.4. Whitelist of SAML Request/Response issuers are now considered holding URIs instead of URLs

This adaptation is motivated by (EID-652) Problem in validation of entityID of SP

Until now, the whitelisted issuers were considered by the eIDAS nodes as being URLs. But this was a not compliant with 8.3.6 Entity Identifier of SAML Core 2.0 specification which states that such an identifier can be defined as URI of no more than 1024 characters.

3.4.1. Code changes

eu.eidas.util.WhitelistUtil

To validate a URI, the WhitelistUtil class is doing 2 things:

1. For URI syntactic verification, it uses the create method from the java URI class. This method verifies if the checked URI violates the RFC 2396 directive.
2. To fully achieve SAML 2.0 compliancy, it also checks that the whitelisted URIs are no more than 1024 characters.

WhitelistUtil logging has also been adapted. For each URI failing validation check a WARN message is printed in the log files with a corresponding stacktrace:

1. For a URI syntax error, the logged Exception is an IllegalArgumentExxception caused by a java.net.URISyntaxException.
2. For a too long URI, the logged exception is an IllegalArgumentExxception with the message: "Non SAML compliant URI. URI is more than 1024 characters in length".

For more information on the java URI class and its validation process, please see the corresponding Javadoc and RFC 2396 directive.

3.5. Revert Interface changes in EIDAS-SAMLEngine related to issuer whitelist logic of a Request/Response and Refactor metadata whitelist from SamlEngine to MetadataFetcher

3.5.1. Code changes

EIDAS-SAMLEngine module

src\main\java\eu\eidas\auth\engine\ProtocolEngineI.java
src\main\java\eu\eidas\auth\engine\ProtocolEngine.java

Removed from the API declared by the interface all parameters related to whitelisting of the issuer metadata URI. Also removed all parameters and the supporting logic in the implementation class.

src\main\java\eu\eidas\auth\engine\configuration\dom\ConfigurationKey.java
ConfigurationKey.METADATA_FETCHER_CONFIGURATION_FILE
New configuration key for the location of the MetadataFetcher

src\main\java\eu\eidas\auth\engine\configuration\dom\ParameterKey.java
ParameterKey.METADATA_FETCHER_CONFIGURATION_FILE
ParameterKey.METADATA_FETCHER_WHITELIST_FLAG
ParameterKey.METADATA_FETCHER_WHITELIST

Added parameters to give access to (SpringManaged)MetadataFetcher (used by the ProtocolProcessor) two configuration properties.

src\main\java\eu\eidas\auth\engine\configuration\dom\DOMConfigurator.java

Added logic to method DOMConfigurator.configureProtocolProcessor() to properly instantiate the MetadataFetcher taking into account the values of the configuration properties listed above.

src\test\java\eu\eidas\auth\engine\ProtocolEngineTest.java
src\test\java\eu\eidas\auth\engine\core\eidas\TestMetadataFetcher.java
src\test\java\eu\eidas\engine\test\simple\AuthResponseTest.java
src\test\java\eu\eidas\engine\test\simple\eidas\EidasMessageFormatOnlyTest.java
src\test\java\eu\eidas\engine\test\simple\eidas\EidasAuthResponseTest.java
src\test\java\eu\eidas\engine\test\simple\eidas\EidasAuthRequestTest.java
src\test\java\eu\eidas\engine\test\simple\eidas\EidasAuthRequestSignatureTest.java
src\test\java\eu\eidas\engine\test\simple\SAMLEngineTimeSkewTest.java
src\test\java\eu\eidas\engine\test\simple\AuthRequestSignatureTest.java
src\test\java\eu\eidas\engine\syntax\EidasSAMLRequestSyntaxTest.java
src\test\java\eu\eidas\engine\syntax\SyntaxTestUtil.java

Adapted the unit tests to the modified/reverted API.

EIDAS-Node module

src\main\java\eu\eidas\node\auth\metadata\WrappedMetadataFetcher.java deprecated as not used

src\main\java\eu\eidas\node\auth\metadata\SpringManagedMetadataFetcher.java

Moved the issuer whitelisting necessary logic to the metadata fetcher from the previous location in EIDAS-SAMLEngine: added constructors, extracted whitelisting logic to WhitelistingMetadataFetcher superclass in the SAML-Metadata module.

src\main\java\eu\eidas\node\auth\connector\AUCONNECTORSAML.java
src\main\java\eu\eidas\node\auth\service\AUSERVICESAML.java
src\main\java\eu\eidas\node\auth\service\ISERVICESAMLService.java

Cleaned up and removed all whitelisting references and logic from these components as unnecessary.

src\test\java\eu\eidas\node\auth\service\tests\AUSERVICESAMLWhitelistTest.java

Removed this test since the class under test (AUSERVICESAML) does not handle anylonger the whitelist logic.

EIDAS-Metadata module

src\main\java\eu\eidas\auth\engine\metadata\impl\WhitelistingMetadataFetcher.java
src\main\java\eu\eidas\auth\engine\exceptions\EIDASMetadataRuntimeException.java

3.5.2. Configuration changes

server\eidas.xml

Removed the entry for key “connector.metadata.location.whitelist”.

New module folder under server, named **metadata,** holding two new external files (one for Connector metadata fetcher one for Service metadata fetcher) to capture the metadata fetcher 2 new config properties
server\metadata\MetadataFetcher_Connector.properties
server\metadata\MetadataFetcher_Service.properties

3.6. Xalan library has been removed from the eIDAS-Node distribution

Since eIDAS-Node 2.2, Xalan was included into EIDAS-Node.war. This was removing the need of a manual installation of the Xalan library in dedicated directories on the different supported application servers. From now on, xalan-2.7.2 jar and all its dependencies (serializer, xerces-impl and xml-apis jar files), is removed from the EIDAS-Node.war. Removing Xalan was needed to get the EIDAS-Node compatible with JAXP 1.5. The compatibility with JAXP 1.5 is ensured by the default XML libraries existing in JDK 7 update 40 and above and, in JDK 8.

3.6.1. Configuration changes

The EIDAS-Node.war file can be deployed as is in glassfish5, Websphere and Weblogic.

JBoss Configuration: JBoss7 is not supported anymore. It was replaced by wildfly versions 15.0.1. Note that the version 11.0.0 is still supported.

Tomcat Configuration: Remove the xalan library and the xerces library from the tomcat lib or shared directory.

Wildfly 11.0.0 Configuration: Add the following line in the paths element of %WILDFLY_HOME%\modules\system\layers\base\sun\jdk\main\module.xml:

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

3.7. Host header poisoning fix for report URI value

This change addresses the vulnerability reported in EID-666:

The server trusts the user-supplied host header. However, by supplying a malicious host header it is possible to:

- change the behaviour of the application;
- modify multiple headers in the response;
- change the domain of the cookie;

- inject content into the body of the response.

3.7.1. Code changes

EIDAS-Node module
src\main\java\eu\eidas\node\security\ConfigurationSecurityBean.java added field and accessor methods for the cspReportingUri that appears in the Node's response headers.

src\main\java\eu\eidas\node\security\SecurityResponseHeaderHelper.java added the logic necessary to build and populate the corresponding response headers.

3.7.2. Configuration changes

Added to eidas.xml the key for `security.header.CSP.report.uri`

3.8. Improved message logging of incoming/outgoing requests/responses

Added new logging entries for the logging of the incoming/outgoing LightRequest/Response and eIDAS messages between eIDAS-Node and the Specific parts and between eIDAS-Connector and eIDAS-ProxyService. The new entries allow for better correlation between messages related to the same eIDAS authentication flow.

3.8.1. Code changes

Several new classes were added:

ConnectorIncomingEidasResponseLogger, ConnectorIncomingLightRequestLogger, ConnectorOutgoingEidasRequestLogger, ConnectorOutgoingLightResponseLogger, ProxyServiceIncomingEidasRequestLogger, ProxyServiceIncomingLightResponseLogger, ProxyServiceOutgoingEidasResponseLogger, ProxyServiceOutgoingLightRequestLogger

Also added the corresponding new servlet filters that intercept incoming/outgoing messages and that use those new Logger classes to log the adequate information:

ConnectorIncomingEidasResponseLoggerFilter,
ConnectorIncomingLightRequestLoggerFilter,
ConnectorOutgoingEidasRequestLoggerFilter,
ConnectorOutgoingLightResponseLoggerFilter,
ProxyServiceIncomingEidasRequestLoggerFilter,
ProxyServiceIncomingLightResponseLoggerFilter,
ProxyServiceOutgoingEidasResponseLoggerFilter,
ProxyServiceOutgoingLightRequestLoggerFilter

From Logger classes, the new class MessageLoggerBean is used to create the appropriate log message for each point.

Also added MessageLoggerUtils which contains several methods involved in the logging. IMessageLogger interface was also added. This interface is implemented by all Logger classes.

For the mentioned new classes, corresponding Junit test classes were added:

ConnectorIncomingEidasResponseLoggerTest,
ConnectorIncomingLightRequestLoggerTest, ConnectorOutgoingEidasRequestLoggerTest,
ConnectorOutgoingLightResponseLoggerTest

ProxyServiceIncomingEidasRequestLoggerTest,
ProxyServiceIncomingLightResponseLoggerTest,
ProxyServiceOutgoingEidasResponseLoggerTest,
ProxyServiceOutgoingLightRequestLoggerTest

ConnectorIncomingEidasResponseLoggerFilterTest,
ConnectorIncomingLightRequestLoggerFilterTest,
ConnectorOutgoingEidasRequestLoggerFilterTest,
ConnectorOutgoingLightResponseLoggerFilterTest

ProxyServiceIncomingEidasRequestLoggerFilterTest,
ProxyServiceIncomingLightResponseLoggerFilterTest,
ProxyServiceOutgoingEidasResponseLoggerFilterTest,
ProxyServiceOutgoingLightRequestLoggerFilterTest

A new class FlowIdCache was added to deal with the flowId functionality (i.e., id used to relate messages from the same eIDAS authentication flow).

A new class SpecificConnectorCommunicationServiceExtensionImpl was added that implements SpecificCommunicationServiceExtension, also new, to be used for exchanging of ILightRequest and ILightResponse between the specific connector and node connector. Similarly, SpecificProxyserviceCommunicationServiceExtensionImpl was added that implements SpecificCommunicationServiceExtension to be used for exchanging of ILightRequest and ILightResponse between the Specific ProxyService and eIDAS ProxyService.

Also added the new test classes:
SpecificConnectorCommunicationServiceExtensionImplTest,
SpecificProxyserviceCommunicationServiceExtensionImplTest.

At EIDASValues, several constants were added, as follows, to be used to fill in the opType field: SPECIFIC_EIDAS_CONNECTOR_REQUEST,
CONNECTOR_SERVICE_REQUEST, EIDAS_SERVICE_SPECIFIC_REQUEST,
EIDAS_SERVICE_SPECIFIC_RESPONSE, EIDAS_SERVICE_CONNECTOR_RESPONSE,
EIDAS_CONNECTOR_CONNECTOR_RESPONSE, COUNTRY_CODE

At NodeBeanNames, several constants were added as follows:

EIDAS_CONNECTOR_FLOWID_CACHE, EIDAS_PROXYSERVICE_FLOWID_CACHE,
CONNECTOR_INCOMING_LIGHT_REQUEST_LOGGER,
PROXY_SERVICE_OUTGOING_LIGHT_REQUEST_LOGGER,
PROXY_SERVICE_INCOMING_LIGHT_RESPONSE_LOGGER,
CONNECTOR_OUTGOING_LIGHT_RESPONSE_LOGGER,
CONNECTOR_OUTGOING_EIDAS_REQUEST_LOGGER,
PROXY_SERVICE_INCOMING_EIDAS_REQUEST_LOGGER,
PROXY_SERVICE_OUTGOING_SAML_RESPONSE_LOGGER,
CONNECTOR_INCOMING_EIDAS_RESPONSE_LOGGER,.

Consequently to this, in the applicationContext.xml the corresponding beans were created: messageLoggerUtils, connectorIncomingLightRequestLogger,
connectorOutgoingLightResponseLogger, proxyServiceOutgoingLightRequestLogger,
proxyServiceIncomingLightResponseLogger, connectorOutgoingEidasRequestLogger,

connectorIncomingEidasResponseLogger, proxyServiceIncomingEidasRequestLogger, proxyServiceOutgoingEidasResponseLogger.

The specificCommunicationDefinitionApplicationContext.xml file was also changed and new beans were added: springManagedSpecificConnectorCommunicationServiceExtension and springManagedSpecificProxyserviceCommunicationServiceExtension.

3.8.2. Configuration changes

A new property was added at server\eidas.xml: saml.audit with a default value set to true. This will be used to activate logging of requests/responses between EidasNode and the Specific.

For server\specificConnector\specificCommunicationDefinitionConnector.xml, two new keys lightToken.connector.request.node.id and lightToken.connector.response.node.id were added, to identify the sender of Light Request and to identify the receiver of the Light Response to/from Connector, respectively.

Similarly at server\specificProxyService\specificCommunicationDefinitionProxyservice.xml were added new keys lightToken.proxyservice.request.node.id and lightToken.proxyservice.response.node.id to identify the receiver of LightRequest and identify the sender of the LightResponse from/to ProxyService.

3.9. Correction for isValidateSignature in ProtocolEngine

The flag isValidateSignature was not correctly used by the code to enabled/disable signature validation. Therefore, a correction was done so that this flag, when set, can in fact decide to perform validation or not of the signature of the Response.

3.9.1. Code changes

The method ProtocolEngine#validateSignatureAndDecryptAndValidateAssertionSignatures was changed to allow disabling the signature validation by configuration when set on the SAML Engine files and continue the flow. Before, and when isValidateSignature was set to false on the SamlEngine_Service.xml file, the response assertions would not be decrypted and the response unmarshall process would fail.

Note that by default, when the flag is not defined, the signature validation is performed as before.

3.10. Upgrade of Bouncy Castle dependency version to 1.60

The version of Bouncy Castle dependency was upgraded from 1.52 to the latest 1.60, due to related CVEs (See section 3.1.1 Bouncycastle CVEs).

3.10.1. Code changes

The EIDAS-Parent/pom.xml was changed, in the following property:

<bouncycastle.version>1.60</bouncycastle.version>

which before had the value of 1.52 and is now 1.60.

3.10.2. Configuration changes

The AdditionalFiles/JBOSS7/ folder was removed and two new folders were added to AdditionalFiles: Wildfly11 and Wildfly15 that contain the Bouncy Castle configuration files to be applied to Wildfly 11.0.0 and 15.0.1 respectively. The bcprov-jdk15on-1.60.jar is a signed version obtained from https://www.bouncycastle.org/download/bcprov-jdk15on-160.jar.

More details on the procedure to be applied can be found in section 6. "Verifying the installation" of eIDAS-Node Migration Guide.

3.11. Removal of Jboss7 related code

Since Jboss7 server is not supported anymore that led to some removal of code.

3.11.1. Code changes

The jBoss7 maven profile was removed from Eidas-node pom.xml as well as the EIDAS-Node/src/main/config/Jboss7 folder.

3.12. Maven profile for wildfly added to allow logging of all files

For Wildlfy deployments it is needed to activate the wildlfy profile, so that the environment variable "LOG_HOME" when set can work. Otherwise, the log files e.g.

eIDASNodeDetail.[date].log

eIDASNodeSAMLExchange.[date].log

eIDASNodeSecurity. [date].log

eIDASNodeSystem. [date].log

will not be created and subsequently the logs will not be there.

3.12.1. Code changes

A new file was added:

/EIDAS-Node/src/main/config/wildfly/jboss-deployment-structure.xml

And a new profile "wildfly" was added to include this file in EidasNode.war.

3.13. Correction supported values in SAML response message

3.13.1. Code changes

Two values for the consent, 'current-implicit' and 'current-explicit', which are used in the internal SAML engine, contained a typo, and thus they were different from the SAML specifications.

The typo was corrected in the DefaultCoreProperties and the EidasResponseValidator. The allowed values for consent attribute where corrected to 'current-implicit' and 'current-explicit' instead of the incorrect 'curent-implicit' and 'curent-explicit'.

All allowed values for the consent are found as a constant in both EidasResponseValidator and EidasAuthnRequestValidator, as: CONSENT_ALLOWED_VALUES.

3.14. Create size limitation for incoming Connector's LightRequest

An implementation was done to restrict the size allowed for incoming Connector's LightRequest.

3.14.1. Code changes

The class eu.eidas.specificcommunication.protocol.impl.SpecificConnectorCommunicationServiceImpl was modified so that getAndRemoveRequest method calls method isInvalid from new class IncomingLightRequestValidator.

In EIDAS-SpecificCommunicationDefinition module the file specificCommunicationDefinitionApplicationContext.xml a new bean "incomingLightRequestValidator" was added that holds the instance of eu.eidas.specificcommunication.protocol.impl.IncomingLightRequestValidator used in already mentioned SpecificConnectorCommunicationServiceImpl# getAndRemoveRequest method.

3.14.2. Configuration changes

In file specificCommunicationDefinitionConnector.xml a new entry to the properties was added:

<entry key="incoming.lightRequest.max.number.characters">65535</entry>

which defines the maximum number of characters of the incoming LightRequest.

3.15. Create size limitation for incoming ProxyService's LightResponse

An implementation was done to restrict the size allowed for incoming ProxyService 's LightResponse.

3.15.1. Code changes

The class eu.eidas.specificcommunication.protocol.impl.SpecificProxyServiceCommunicationServiceImpl was modified so that getAndRemoveResponse method calls method isInvalid from new class IncomingLightResponseValidator.

In EIDAS-SpecificCommunicationDefinition module the file specificCommunicationDefinitionApplicationContext.xml a new bean "incomingLightResponseValidator" was added that holds the instance of eu.eidas.specificcommunication.protocol.impl.IncomingLightResponseValidator used in already mentioned SpecificProxyServiceCommunicationServiceImpl# getAndRemoveResponse method.

3.15.2. Configuration changes

In file specificCommunicationDefinitionProxyService.xml a new entry to the properties was added:

<entry key="incoming.lightResponse.max.number.characters">65535</entry>

which defines the maximum number of characters of the incoming LightResponse.

3.16. Allow setting of Consent attribute value from configuration files

All different values for the consent attribute can now be chosen from the configuration files:

SamlEngine_Connector.xml
SamlEngine_Service.xml

The current possible values are; 'obtained', 'prior', 'current-implicit', 'current-explicit', 'unspecified', 'unavailable', 'inapplicable'.

3.16.1. Code changes

In the DefaultCoreProperties class, both the loading of the consent authentication request and –reponse were updated to allow all current possible values. In addition; should a blank value be present, it is treated as a null value.

3.16.2. Configuration changes

In both SamlEngine_Connector.xml and SamlEngine_Service.xml the comments describing the possible values were updated for the entries with key 'consentAuthnRequest' and 'consentAuthnResponse'.

3.17. Removal of old audit SAML messages in EidasNode

The previous audit implementation was removed, since the new message logging was implemented and therefore it was unnecessary to maintain it.

3.17.1. Code changes

Deprecated IEIDASLogger. Removed EidasLoggerBean class. Replaced call to toString by call to createLogMessage.

Removed loggerBean field and corresponding getters/setters from AUCONNECTORSAML and AUSERVICESAML.

Removed prepareReqLoggerBean, prepareRespLoggerBean, saveLog methods as well as its usages from AUCONNECTORSAML and AUSERVICESAML.

Removed unused parameters from methods checkAntiReplay and checkResponseLoA from AUCONNECTORSAML.

Removed spring beans ServiceLogger, ConnectorLogger and its references in properties loggerBean from beans springManagedAUCONNECTORSAML and springManagedAUSERVICESAML at applicationContext.xml.

Removed IEIDASLogger mock and call to its setLoggeBean method from a tests AUCONNECTORSAMLTestCase and AUSERVICESAMLTestCertif. Updated copyrights. Removed unused exceptions.

3.18. Java 8 migration

The code is now supporting Java 8 instead of Java 7, which was also aligned with a change in the supported servers. Please check the eIDAS-Node Migration Guide for futher details on the supported servers in 2.3.

3.18.1. Code changes

It was necessary to update some Javadoc in order to resolve several errors occurred when generating the Javadocs. Java 8 is more restricting on this topic than Java 7.

3.19. Other fixes/improvements requiring no action

3.19.1. Dependency of EIDAS-SpecificCommunicationDefinition Junit tests on environment variables set for deployment

In EIDAS-SpecificCommunicationDefinition, Junit tests, depended on eIDAS environment variables being set to be executed successfully. This was modified so that Junit tests can be independent on the environment. The file test/resource/specificCommunicationDefinitionApplicationContext.xml used by Junit tests was modified to point to use a newly added config folder in test resources, with the necessary files to run the tests independently. Also a new file specificCommunicationDefinitionEnvironmentContext.xml was added.

3.19.2. EIDAS-SpecificCommunicationDefinition wrong jar name

The jar produced at the EIDAS-SpecificCommunicationDefinition module had the wrong name. The wrong finalname element was removed from the pom so that the eidas-specific-communication-definition.jar is produced as expected.

3.19.3. Duplicate Copyright Headers removed

In around 150 source files, there was a duplicate copyright header. They were replaced by a single copyright header.

3.19.4. Removal of unused dependencies

See section 3.1.3 JQuery CVEs

Removed jquery-1.11.3.min.js, function.js, switchery.min.css, switchery.min.js from EIDAS-Node; not used within the module.

3.19.5. Removal of no-longer used inclusion files

Removed footerScripts.jsp from the EIDAS-node module, thus also no longer present in error.jsp,

presentError.jsp, presentSamlResponseError.sp, connectorRedirect.jsp, tokenRedirectMsConnector.jsp, tokenRedirectMsProxyService.jsp

3.19.6. Upgrade bootstrap to v4.3.1

The bootstrap.min.js and bootstrap.min.css files were upgraded to version 4.3.1(See section 3.1.2 Bootstrap CVEs)

3.19.7. Update of EIDAS Metadata VERSION 2.3.0

The eidas.application.identifier value was updated to CEF:eIDAS-ref:2.3.

3.19.8. Created Junit Test for existing code

More Junit test classes and methods were created to improve the coverage and also check that potential vulnerabilities could not be exploited, such as: XmlSchemaUtilTest, DocumentBuilderFactoryUtilTest.

3.19.9. Sanitized input/output fields

To prevent possible cross-site scripting, an additional number of input- and outputfields where sanitized in the following jsp-files; colleagueRequestRedirect.jsp, connectorRedirect.jsp, tokenRedirectMsConnector.jsp, tokenRedirectMsProxyService.jsp.

3.19.10. Disabling DTD in the XML parser for incoming LightRequest

This adaptation is motivated by (EID-914) LightRequest and LightResponse exposed to DTD threats.

The method unmarshallRequest in class eu.eidas.specificcommunication.protocol.impl. LightJAXBCodec was improved so that an incoming LightRequest with DTD is not allowed.

3.19.11. Disabling DTD in the XML parser for incoming LightResponse

This adaptation is motivated by (EID-914) LightRequest and LightResponse exposed to DTD threats.

The method unmarshallResponse in class eu.eidas.specificcommunication.protocol.impl. LightJAXBCodec was improved so that an incoming LightResponse with DTD is not allowed.

3.19.12. Fixing build failure due to JUnit failure on LINUX

Changes were done to test classes

EIDAS-Node/src/test/java/eu/eidas/node/auth/metadata/TestEidasNodeFileMetadataProcessor.java

EIDAS-Node/src/test/java/eu/eidas/node/auth/metadata/TestEidasNodeFileMetadataProcessorTrustChain.java

EIDAS-Node/src/test/java/eu/eidas/node/auth/metadata/TestEidasNodeMetadataLoader.java

EIDAS-Node/src/test/java/eu/eidas/node/auth/TestEidasNodeMetadataTrustChain.java

EIDAS-Node/src/test/java/eu/eidas/node/auth/ metadata/util/tests/FileUtils.java

to fix the failing tests due to Error initializing the test environment originated from method initWorkFolder.

3.19.13. Use of UTC (Zulu) format in the logs of EidasNode

In logback.xml it was replaced the pattern of log entries and of log filenames to be GMT.

3.19.14. Generate a new Id for the outgoing LightResponse.

A change was done in ColleagueResponseServlet to generate a new Id to outgoing LightResponse other than the related eIDAS Response.

3.19.15. Generate a new Id for the outgoing LightRequest.

A change was done in ColleagueRequestServlet to generate a new Id to outgoing LightRequest other than the related eIDAS Request.

3.19.16. Validate entityId from metadata files against illegal characters.

A new validation has been added to check the entityId string from metadata files against illegal characters. EIDASMetadataProviderException exception will be thrown if entityId has illegal characters;