eIDAS-Node Migration Guide

For Version 2.5

Document history

Version	Date	Modification reason	Modified by
1.0	11/12/2020	Information how to migrate from eIDAS-Node v2.4 to eIDAS-Node v2.5	DIGIT

© Copyright European Commission —

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.

© European Union, 2020

Reuse of this document is authorised provided the source is acknowledged. The Commission's reuse policy is implemented by Commission Decision 2011/833/EU of 12 December 2011 on the reuse of Commission documents.

Table of contents

DOCUMENT HISTORY 2
TABLE OF CONTENTS 4
1. INTRODUCTION 8
1. Document structure 8
2. Document aims 8
2. PREREQUISITES 9
3. CHANGES 10
1. Summary of changes 10
2. Metadata cache 11
Code changes 11
3. Externalize Specific-Communication caches names 11
Code changes 11
Configuration changes 12
4. Refactoring of Loggers: Centralization of the utils methods 12
Code changes 12
5. Change how communication cache message is send to loggers 12
Code changes 12
6. Introduction of the Full Message Logging 13
Code changes 13
Configuration changes 14
7. Node Country code publication in metadata 14
Code changes 14
Configuration changes 15
8. Restrict TLS version used by the Node to TLS v1.2 15
Code changes 15
Configuration changes 15
9. Signature algorithm alignment with technical specifications 1.2. 15
Code changes 15
Configuration changes 16
10. Publishing of Non-Notified Levels of Assurance 17
Code changes 17
Configuration changes 17
11. Extension of the LightRequest for the RequesterId 18
Code changes 18
Configuration changes 18
12. Forward the requesterId from Light Request into the eIDAS Request 18
Code changes 19
13. Forward requesterId from eIDAS Request into Light Request 20
Code changes 20
14. Validation of the RequesterId in the Connector 20
Code changes 20
15. Validation of the requesterId in the proxyService 21

© Copyright European Commission —

16. Add RequesterID flag to Proxy-Service's metadata 21
Code changes 22
Configuration changes.....22
17. RequesterID is forwarded from Connector to ProxyService only when SPType is private.....23
Code changes 23
18. Remove validation of ProviderName as not mandatory 23
19. Replace Glassfish 5 full platform support by Glassfish 5.0.1 Web profile .24
20. Adaptation of the eidas.protocol.version from 1.1 to 1.2.....24
Configuration changes.....24
21. Publishing custom list of eIDAS attributes at ServiceMetadata 24
Code changes 24
Configuration changes.....25
22. Extension of the LightResponse with Consent and setting the consent in the eIDAS SAML Response.....25
Code changes 25
23. Extension of the LightRequest with spCountryCode and removing the option to forward spCountryCode via citizenCountryCode 25
Configuration changes.....26
24. Level of Assurance Validation for Eidas Request and Eidas Response.....26
Code changes 26
25. Level of Assurance Type Validation for Light Request 27
Code changes 28
26. Validation for published Levels of Assurance on the Proxy Service 28
Code changes 28
27. Digest method algorithm configuration.....28
Code changes 28
Configuration changes.....29
28. Fix digest algorithm list in Node metadata and preference order of algorithms 29
Code changes 29
29. Validation of the response in representation cases 29
Configuration changes.....30
30. Remove unnecessary validation of connector destination url 30
Code changes 30
Configuration changes.....30
31. Connector validation for NameID format.....30
Code changes 30
Configuration changes.....31
32. Generation of LightRequest LightResponse Classes from XSD.....31
Code changes 32
Configuration changes.....32
33. Other fixes/improvements requiring no action 32
CVE-2018-10237 32
CVE-2020-1963.....32
CVE-2019-12400 32

© Copyright European Commission —

CVE-2015-0254 32
Upgrade of se.swedenconnect.opensaml:opensaml-security-ext from 1.0.5 to 1.0.7 33
Update eIDAS Metadata VERSION 2.5.0 Application Identifier 33
Node country consume from metadata 33
Support for multiple application identifier values in Metadata 33
Support for multiple protocol versions AttributeValue in Metadata 33
Metadata retrieval read for active.module.connector or active.module.service are false 34
Flow fails with Invalid Light Response error if the connector metadata is not reachable 34
Extension of the LightRequest for NonNotified Level of Assurance 34
Extension of the EidasAuthenticationRequest for Non-Notified Levels of Assurance 35
Forward of Non-notified Levels of Assurance from EIDAS Request to Light Request 36
Update the validation of the Levels of Assurance 36
Allow validation for non-notified LoA in proxyservice 36
Validation of Levels of Assurance in Light Request Builder 36
Level of Assurance backwards compatibility with EIDAS protocol version 1.1 37
Response Level of Assurance Correlation with Service Metadata Levels of Assurance. 37
Log the complete Light Response received by the Proxy-Service 38
Log the complete Light Request received by the Connector 38
Truncated messages logged in errors based on maximum message length 38
Modification of logging processing 38
SAML Response message is exposed in the logs when sending big light response 39
Add validation for the key length of the certificates used to sign metadata and authentication request 39
Add validation of the signature algorithm's hash length of the certificates used to sign the metadata and/or authentication request 39
Implementation to determinate digest algorithm 40
Update EUID attribute validation 40
Update value and validation of Gender possible values based on protocol version 40
Citizen country code validation 41
Connector check the nationality code of a sending country in assertions 41
Update VersionMismatch as StatusCode instead of SubStatusCode in LightResponse.xsd 42
Validation of StatusCode and Substatus code values 42
XML for failed LightResponse is not valid according XSD 42
Externalised maximum parameter length for EIDAS response/request validation 42

© Copyright European Commission —

Discontinue support for sys property "org.opensaml.httpclient.https.disableHostnameVerification" 42
Remove config parameter "include.assertion.fail.response.application.identifiers" 43
Fix ProxyService Metadata generator usage of Metadata enabled flag.....43
Removed Hazelcast from EIDAS-Commons 43
Update SimpleProtocol version in EIDAS-Parent dependency management 43
Commented code removed from Application Context xml files 44
Removed initialization of SubStatusCode in ResponseStatus factory method.....44
Introduction of MessageLog model to replace StringBuilders in Loggers..44
Fix incorrect constant value (cfr. (EID-636) Question on "EIDAS SAML Engine 1.4.1") 45
Fix duplicate filtering of Outgoing EidasResponse in ProxyService 45
Fixed concurrency issue in LoggingConcurrencyTest.....45
Javadoc improvement: Reduce Javadoc warnings 45
(EID-1051) Fix incorrect usage of retrieveConnectorAttributes 45
Creation of an default eidas.xml configuration file.....45
Avoid Ignite to make http request to verify if version is up-to-date.....46
Transform EIDAS-Config properties into a deploy-able set of property files.....46
Inconsistencies in Eidas-attributes.xml config files 46
Fix non deterministic build failure build of tests in EIDAS-SAMLEngine Module 46
Javadoc is pointing to itself.....46
Unit tests performance improvements.....46
Remove rsa-ripemd160 from test code.....47
Unit test failure on UNIX Cloud environment 47
Sonar: EIDAS-SpecificCommunicationDefinition: new vulnerability reported (minor).....47
CertificateUtilTest class refactoring and improvements 47
SAMLAuthnResponseDecrypterTest improvements.....47
Regression: 'testConfiguredCacheMultipleExpiringKeys' unit test failure..47
Validation of LightRequest received by the Connector.....47
(EID-1139) Invalid omitXmlDeclaration 48
Removed decodeCurrentAddress javascript function.....48
(EID-1136) Typos “metdata” instead of “metadata” 48
(EID-1115) Wrong error message in XmlSchemaUtilTest 48
(EID-1135) - Typos "fecther" instead of "fetcher" 48
(EID-1119) – eIDAS-Node Error Codes has not deprecated codes.....49
Upgrade org.apache.httpcomponents:httpclient to version 4.5.13.....49

© Copyright European Commission —

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 from eIDAS-Node v2.4 to eIDAS-Node v2.5.

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 2.5.

Chapter 3 — *Changes*: Contains detailed information about the changes that should be taken into consideration when migrating to eIDAS-Node version 2.5.

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 2.5, including:

- configuration changes; and
- changes to code.

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. Prerequisites

Before starting your migration to eIDAS-Node version 2.5 you should have:

- already implemented eIDAS-Node version 2.4;
- downloaded the eIDAS-Node v2.5 Integration Package; and
- downloaded the latest documentation.

3. Changes

1. Summary of changes

(EID-636) Question on "EIDAS SAML Engine 1.4.1"

(EID-974) Signature algorithms listed as digest algorithm in metadata

(EID-978) XML for failed LightResponse is not valid according XSD

(EID-980) VersionMismatch in LightResponse should be statusCode and not subStatusCode

(EID-1033) - CVE-2015-0254

(EID-1035) - CVE-2018-10237

(EID-1043) - Avoid or document that Ignite and other entities "call home"

(EID-1051) - Method 'retrieveProxyServiceAttributes' possibly removed by mistake

(EID-1057) - Introduction of consent

(EID-1063) - Before validating the signature of a SAML object, the node does not check the digest algorithm

(EID-1081) - CVE-- 2020-1963

(EID-1105) - Unknown hash algorithm null

(EID-1110) – Node 2.5.0 produce illegal Metadata with schema violation

(EID-1112) - Unsupported algorithms shouldn't be allowed in metadata7

(EID-1115) - Wrong error message in XmlSchemaUtilTest

(EID-1118) - websphere session cookie configuration rejected

(EID-1119) - eIDAS-Node Error Codes document has not deprecated codes

(EID-1121) - TLS configuration in documentation still mentions TLSv1.1

(EID-1123) - Documented Websphere configuration context roots are incorrect

(EID-1127) - Application server folder names incorrect in documentation

(EID-1128) - Deployment procedures inconsistency

(EID-1130) - Incorrections in eIDAS-Node Installation and Configuration Guide document

(EID-1131) - The XML LightRequest figure missing in eIDAS-Node National IdP and SP Integration Guide document

(EID-1132) - Incorrections in eIDAS-Node and SAML document

© Copyright European Commission —

(EID-1134) - Broken reference in eIDAS-Node Migration Guide 2.5 documentation

(EID-1135) - Typos "fecther" instead of "fetcher"

(EID-1136) - Typos "metdata" instead of "metadata"

(EID-1138) - JavaScript error in citizenConsentResponse.jsp page

(EID-1139) - Invalid value "false" for omitXmlDeclaration

(EID-1142) - CVE-2020-13956

2. Metadata cache

Metadata caches, distributed or not, now respond to the refresh of the application context by clearing their key-value pairs.

Code changes

The interface IClearableCachingService was introduced to provide a shared interface between implementations in function of clearing the cache. As a result the 3 implementation modules:

- EIDAS-JCache-Dev-Node,
- EIDAS-JCache-Ignite-Node,
- EIDAS-JCache-Hazelcast-Node

have changed their

- AbstractMetadataCaching (implementation)
- AbstractMetadataCachingTest (coverage)

ReloadMetadataCacheAppContextListener was introduced to hook clearing the services after ApplicationContextRefresh and is defined in applicationContext.xml

3. Externalize Specific-Communication caches names

Names of both Ignite and Hazelcast caches were externalised and can be optionally set in specificCommunicationDefinitionConnector.xml and specificCommunicationDefinitionProxyService.xml and also in igniteSpecificCommunication.xml or hazelcastSpecificCommunication.xml depending if it is Ignite or Hazelcast being used. The references for the Hazelcast caches were adapted so that both cache types can be set through these files.

Code changes

In EIDAS-SpecificCommunicationDefinition the specificCommunicationDefinitionApplicationContext.xml was modified to read properties from default files specificCommunicationDefinitionConnector.xml and specificCommunicationDefinitionProxyservice.xml.

A new test class SpecificDefinitionApplicationContextTest.java was added to test the external and default cache name setting.

Configuration changes

The following configuration files were changed:

hazelcastSpecificCommunication.xml

- Default name for cache specificNodeConnectorRequestProviderCacheService was changed to specificNodeConnectorRequestCache
- Default name for cache nodeSpecificProxyserviceRequestProviderCacheService was changed to nodeSpecificProxyserviceRequestCache
- Default name for cache specificNodeProxyserviceResponseProviderCacheService was changed to specificNodeProxyserviceResponseCache
- Default name for cache nodeSpecificConnectorResponseProviderCacheService was changed to nodeSpecificConnectorResponseCache

4. Refactoring of Loggers: Centralization of the utils methods

Code changes

The AbstractLogger and the MessageLoggerTag classes were removed as not used after the centralization of the util methods and after the modifications made for the MessageLog Model (see “32.45 Introduction of MessageLog model to replace StringBuilders in Loggers”).

Util methods 'createMsgHash' and 'createBltHash' were added in the LoggingUtil class, and the Logger implementations were adapted to use these new methods.

The Logger tests classes were also adapted to work correctly with these changes.

5. Change how communication cache message is send to loggers

The loggers for the specific communication formerly used to pop, process and push messages of the cache. Since there is nothing in the process for loggers, this was replaced by a peek, not manipulating the cache.

Code changes

A new interface SpecificCommunicationLoggingService has been created to expose the 'getRequest' and 'getResponse' methods, both having overloads to return the object or the string value, to enable the retrieval of the LightRequest and LightResponse from the caches without removing them.

Two implementations have been made to the SpecificCommunicationLoggingService interface, “SpecificProxyserviceCommunicationLoggerServiceImpl”, ”SpecificConnectorCommunicationLoggerServiceImpl”.

© Copyright European Commission —

Implementation of the loggers for Light request / Light response Loggers have been updated replacing the 'getAndRemove + put' for the 'getRequest' so that the request/response is not removed.

"NodeId" has been added to the bean of some loggers where is was used to create "NodeId" in the Specificommunication Logs. This to replace the direct use of a service implementation.

Implementation of these changes is done in the Abstract class "AbstractSpecificCommunicationLoggingService", which is further implemented for Connector and ProxyService in "SpecificConnectorCommunicationLoggerServiceImpl" and "SpecificProxyserviceCommunicationLoggerServiceImpl" respectively.

Unit test have been adjusted accordingly.

- SpecificConnectorCommunicationLoggerServiceImplTest
- SpecificProxyserviceCommunicationLoggerServiceImplTest

6. Introduction of the Full Message Logging

A functionality for logging of the full content of the incoming or outgoing message (LightRequest, LightResponse, eIDAS Request and eIDAS Response) was implemented for debugging purposes.

Code changes

The logback.xml configuration package with the Node has been updated introducing a new log file appender and therefore a new logging file eIDASNodeFullMsgExchange, where the full content of the requests/responses will be logged.

In order to filter the content that will be logged by this new appender, a new Marker (*FULL_MESSAGE_EXCHANGE*) has also been introduced.

The MessageLoggerUtils class provide a new method to check if the full message logging is active or not.

A new interface (IFullMessageLogger) is created to define a method contract for the full message logging. This interface is implemented by the following loggers:

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

The implementation is logging the full content of the LightRequest, LightResponse, eIDAS Request and eIDAS Response in the newly introduced logback configuration using the new Marker. Validation was added to both Connector and ProxyService to ensure SAML messages will only be logged when the SAML scheme is valid.

© Copyright European Commission —

This implementation is called from the already existing filters matching each incoming or outgoing request/response.

Messages coming from classes implementing IFullMessageLogger are using the logger with name="eu.eidas.logging.full" as reflected in logback.xml and EIDASValues.EIDAS_PACKAGE_LOGGING_FULL.

Configuration changes

Introduction of support for a new optional configuration entry ("full.message.logging") in the eidas.xml to enable or disable the full message logging.

7. Node Country code publication in metadata

Code changes

Following classes have been defined to describe the new NodeCountryType element:

- NodeCountryType
- NodeCountryTypeImpl
- NodeCountryTypeBuilder
- NodeCountryTypeMashaller
- NodeCountryTypeUnmarshaller

EidasMetadataParametersI is now defining 2 new methods:

- getNodeCountry
- setNodeCountry

These new methods are implemented in the EidasMetadataParameters class.

The EidasMetadataGenerator has also been updated in order to publish the NodeCountryType element in the metadata.

The NodeCountry element is either added in the Extensions element of the SPSSODescriptor or in the IDPSSODescriptor of the eIDAS metadata, depending on the role of the node instance (Connector or Proxy-service).

Support for the NodeCountryType has been added to the XMLObjectProviderRegistrySupport within the 'configureExtension' method of the EidasExtensionConfiguration.

A test to verify the validation of the NodeCountry value while publishing the metadata has also been added.

In the context of EID-1110, Node 2.5.0 produce illegal Metadata with schema violation, some modifications were applied to the EidasMetadata class in order to produce valid metadata xml.

The EidasMetadataValidationTest class was added in order to test the xml validity of the generated metadata. The xsds necessary for the validation of the SAML metadata were therefore also added to the test resources. The xsds files are the following:

- saml-schema-assertion-2.0.xsd
- saml-schema-metadata-2.0.xsd
- xenc-schema.xsd
- xmldsig-core-schema.xsd
- xml.xsd

Configuration changes

New property "metadata.node.country" in the eidas.xml to specify the Node country code. According to the specifications, this is a mandatory configuration property.

The Node country code value must be compliant with the ISO 3166-1 alpha-2 format.

8. Restrict TLS version used by the Node to TLS v1.2

In the context of the alignment with the specifications 1.2 of the Eidas Node, the support for TLS v1.1. has been removed.

Code changes

The DEFAULT_TLS_ENABLED_PROTOCOLS variable in the BaseMetadataFetcher class has been adapted to only accept TLSv1.2.

Configuration changes

The "tls.enabled.protocols" property in the eidas.xml configuration file has also been adapted to the unique value of TLSv1.2.

The "tls.enabled.ciphers" parameter was also updated to the list of ciphers suites that should be used by the Node in regards with specifications 1.2. Therefore, the SHA1 ciphers suites were removed from the list of ciphers suites since it is not present in the table of ciphers of the specifications.

9. Signature algorithm alignment with technical specifications 1.2.

In the context of the alignment with the specifications 1.2 of the Eidas Node, the list of Signature algorithms has been adapted.

Code changes

The AbstractProtocolSigner class has been modified to remove two following private constants:

- ALLOWED_ALGORITHMS_FOR_SIGNING
- ALLOWED_ALGORITHMS_FOR_VERIFYING

These two constants are not used anymore, since the specifications are now even restrictive for signing and for verifying. Therefore, validation of signing signatures or verifying signatures are based on the default Signature algorithms list which has been updated with the values from the specifications.

The default signature algorithm has also been changed since the previous value of default signature algorithm is no longer part of the allowed signature algorithms.

The new value of the default signature algorithm is:

http://www.w3.org/2007/05/xmldsig-more#sha512-rsa-MGF1

But, for interoperability reasons it was overridden in the external eidas.xml by

http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1

since this value is both valid in previous and new eIDAS Node versions signature algorithms list.

The signature.algorithm, to allow compatibility with previous versions of the Node, should be one of the four next values:

- http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1
- 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

Note that this was agreed on consultation page at https://ec.europa.eu/cefdigital/wiki/display/EIDTECHSUB/02.+Signature+algorithms+for+SAML+and+SAML+Metadata

Tests in EidasAuthRequestSignautreTest class and in TestEidasNodeMetadataLoader class have been adapted in accordance to the changes.

Configuration changes

Multiple changes to the configuration have been done.

The following configuration files have been modified:

- EIDAS-Config/server/eidas.xml
  The signature.algorithm property was changed to http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1 in order to override the default

- EIDAS-Config/server/SignModule_Connector.xml
  EIDAS-Config/server/SignModule_Service.xml

  The signature.algorithm value was changed to http://www.w3.org/2007/05/xmldsig-more#sha512-rsa-MGF1

- EIDAS-Config/server/SignModule_Connector_EC.xml
- EIDAS-Config/server/SignModule_Service_EC.xml

- The signature.algorithm value was changed to http://www.w3.org/2007/05/xmldsig-more#ecdsa-sha512

The "signature.algorithm.whitelist" property entry has been changed in eidas.xml file as follows:

```
<entry key="signature.algorithm.whitelist">
   http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1;
   http://www.w3.org/2007/05/xmldsig-more#sha384-rsa-MGF1;
   http://www.w3.org/2007/05/xmldsig-more#sha512-rsa-MGF1;
   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
</entry>
```

Note, that even if algorithms, other than the ones in the specification, are set in this entry's values, those will not be published in metadata pages.

In other files, SignModule_YYYYY.xml files, it was removed since it was overridden by eidas.xml.

10. Publishing of Non-Notified Levels of Assurance

In the context of the implementation of the technical specifications 1.2, the Proxy-Service needs to be able to publish multiple Levels of Assurance.

Code changes

The EidasMetadataParametersI interface was changed to introduce new methods to retrieve the list of Levels of Assurance.

The EidasMetadataParameters implementation has been changed to implement the new methods and adapt previous implementation to handle the possible multiple values of the Levels of Assurance.

The EidasNodeMetadataGenerator has been modified to be able to publish multiple values of Level of Assurance in the ProxyService metadata.

The EidasMetadata class was also adapted to be able to parse all the values of the Levels of Assurance that would be published in the Node metadata.

The EidasNodeValidationUtil class was also modified to update the 'isRequestLoAValid' implementation to accept non-notified levels of Assurance also.

Unit tests of the EidasNodeValidationUtil in EidasNodeValidationUtilTestCase have been updated and others have been added.

Configuration changes

The configuration parameter "service.LoA" can now have multiple values which need to be added as a comma (or semi-colon) separated list.

Note that for interoperability reasons with older versions, the highest notified LoA has to be the first value in parameter "service.LoA".

Note that now, all supported notified and non-notified LoAs, have to be published. This is due to 1.2 eIDAS specification. In previous nodes, which followed 1.1 eIDAS specification, only the Highest notified LoA was to be published.

© Copyright European Commission —

11. Extension of the LightRequest for the RequesterId

In the context of the alignment of the code of eIDAS node with the Technical Specifications v1.2, Service Providers will have to provide a RequesterId.

The requester id is a unique (inside a member state) identifier for Service Providers.

Therefore, a new element "requesterId" has been added to the LightRequest.

This new element is an optional element so the node can still interoperate with previous versions.

Code changes

The following files were changed:

- AbstractAuthenticationRequest.java: requesterId added to class builder
- AbstractLightRequest.java: requesterId added to properties and class builder
- ILightRequest.java: the method 'getRequesterID' was added.
- Unit test for Light Request has been updated
- LightRequest.xsd, resource for Light Request Unit test, was updated to add the new requesterId element

Configuration changes

The LightRequest will now include the requesterId element, which has the value of the requester.id parameter from sp.properties if this parameter is set. To ensure interoperability with versions 2.4 or earlier, this field is optional.

12. Forward the requesterId from Light Request into the eIDAS Request

At the eIDAS Connector, several changes were done in order to forward the requesterId from the Light Request into eIDAS Request.

The eIDAS SAML request sent by the Connector, will include element Scoping, which, by its turn, will include element RequesterId according to the following cases:

Case 1:
If Connector is public and if RequesterId Flag is set in the Proxy-Service metadata
then
RequesterId value in the Light Request is assumed as from a public SP
and it will not be sent in the eIDAS Request from Connector to Proxy-Service

Case 2:
If Connector is private and if RequesterId Flag is set in the Proxy-Service metadata
then
RequesterId value must be present in the Light Request, must not be empty and it is assumed as from a private SP
and it will be sent in the eIDAS Request from Connector to Proxy-Service

Case 3:
If the Connector serves Public and Private SPs and if RequesterId Flag is set in the Proxy-

Service metadata
then
SPtype in Light Request must be not empty and either be set to public or private
and
if SPtype in Light Request is set to public
then RequesterId value is assumed as from a public SP
and will not be sent in the eIDAS Request to Proxy-Service
else if SPtype in Light Request is set to Private
then RequesterId value is assumed as from a private SP, must be present in the Light Request, must not be empty
and will be sent in the eIDAS Request from Connector to Proxy-Service

As decided in https://ec.europa.eu/cefdigital/wiki/display/EIDTECHSUB/04.+RequesterId

Note that, the responsibility of sending the requesterId of Private SPs to the Connector is on the MS side, in the Demo application it was implemented at the Demo Service Provider / Demo Specific Connector.

Several tests were added at AUCONNECTORTest.

Code changes

EidasProtocolProcessor class

Implemented support to RequesterId in method 'marshallRequest'. Logic to send RequesterId when not empty and when ProxyService metadata has RequesterId flag set was added. New methods to build Scoping, RequesterId and include RequesterId in Scoping were also added.

In EidasProtocolProcessorTest class

New test methods: 'marshallRequestNOtEmptyRequesterIdWithFalseRequesterIdFlag', 'marshallRequestNOtEmptyRequesterIdWithFalseRequesterIdFlag', 'marshallRequestNotEmptyRequesterIdWithFalseRequesterIdFlag', 'marshallRequestEmptyRequesterIdWithFlaseNotSetRequesterId', 'marshallRequestWithMetadataFetcherThrowsException', 'unmarshallRequestRequesterIdNotEmpty',

'unmarshallRequestRequesterIdNotPresent' were added.

Added new instances ServiceRequesterId and ServiceMetadataFetcherThrowsException to test SamlEngine.xml with related metadata new files TestMetadataFetcherRequesterId.java and TestMetadataFetcherThrowsException.java and configuration files MetadataFetcher_ServiceMetadataFetcherThrowsException.properties and MetadataFetcher_ServiceRequesterId.properties to support the tests.

EidasMetadata class:

- Added constants ENTITY_CATEGORY_ATTRIBUTE_NAME REQUESTER_ID_FLAG_VALUE.

In MetadataUtil class added support to RequesterId at 'convertExtensions' method, it was created an auxiliary method 'hasRequesterIdValue'.

MetadataUtilTest class:

Two new tests were added, 'testConvertEntityDescriptorWithRequesterIdFlag' and 'testMarshallMetadataParameters' and a new method extracted loadEntityDescriptorsFromFolder'.

Existing test method 'testMarshallMetadataParameters' was refactored to adapt to the changes in the new tests and missing Javadoc was added.

Added new test file ServiceMetadataWithRequesterId.xml with RequesterId flag set.

13. Forward requesterId from eIDAS Request into Light Request

At the eIDAS Proxy-Service, several changes were done in order to forward the requesterId from the eIDAS Request.

Code changes

Some modifications were applied to the EidasProtocolProcessor class to forward the RequesterId received in the SAML EidasRequest into the LightRequest, new method 'extractRequesterId' was added.

Several tests were added in EidasProtocolProcessorTest: 'testGetMetadataProtocolVersions', 'testGetMetadataProtocolVersionsEmpty', 'unmarshallRequestRequesterIdNotPresent'.

14. Validation of the RequesterId in the Connector

With the introduction of the RequesterId for the alignment with the technical specifications 1.2, some validations of the RequesterId presence and value had to be added in the Connector code.

Code changes

Therefore, a new error code and message have been added to be displayed when the RequesterId is missing or its value is invalid when it is required by the destination ProxyService.

For this new error code and message, the following files have been modified:

- EidasErrorKey.java in EIDAS-Common module
- eidasErrors.properties in EIDAS-Common module
- errors.properties in EIDAS-Node module
- sysadmin.properties in EIDAS-Node module

A new method was added to the AUCONNECTORUtil class to get the SP type configuration of the Connector.

The AUCONNECTOR class was modified to use this new method instead of an alternative way of getting that value.

© Copyright European Commission —

The AUCONNECTORSAML class was modified to integrate the validation of the RequesterId which depends on the flag concerning the requesterId given by the destination ProxyService metadata and the SpType of the Connector or request.

The RequesterId cannot be left empty when the SpType is private and the ProxyService requesterId flag is active and if the RequesterId is given (even if not mandatory) it needs to be valid.

The requesterId value should be an URI and should not exceed 1024 characters (cfr SAML core specifications) to be valid.

Therefore, a new constant was added to the EidasParameterKeys class and a new "max.requesterId.size" property was added in the eidasParameters.properties file.

15. Validation of the requesterId in the proxyService

According to the technical specifications v1.2, the requesterId must be present if the request is made by a Private Sector relying party and the ProxyService is exposing the RequesterId flag.

This validation has been added to the AUSERVICE class returning a SAML error message when requesterId is missing when actually needed.

Some unit tests were also added in the AUSERVICETest class to cover up the multiple possible scenarios with and without requesterId in the request.

A new EidasErrorKey COLLEAGUE_REQ_MISSING_REQUESTER_ID has been added with the value 'colleagueRequest.missing.requesterID' to describe the error when the requesterId is missing while required.

The requesterId value must be an URI and must not exceed 1024 characters (cfr SAML core specifications) to be valid.

Note that in order to remain interoperable with older versions of the Node, if the connector is displaying in its metadata that it doesn't comply with a higher technical protocol version than the 1.1, then the request will be accepted even without a requesterId (even if needed, since the requesterId was only added in the specifications 1.2)

16. Add RequesterID flag to Proxy-Service's metadata

Due to 1.2 eIDAS specifications, the Requester ID flag is now possible at the Proxy-Service's metadata. This flag indicates to the Relying party that it must send the Requester Id of Service Provider in the case this one is a private one.

Code changes

The following files were changed/added:

Production code:

EidasMetadata.java (added method 'generateRequesterIdFlagAttribute' that generates entity category attribute for the requester Id flag, updated method

'generateEntityAttributes' to add requesterIdFlagAttribute if requesterIdFlag is set to true)

EidasMetadataParametersI.java (added setter/getter for RequesterIdFlag)

EidasMetadataParameters.java (implemented setter/getter from interface EidasMetadataParametersI)

EidasNodeMetadataGenerator.java (added method 'generateRequesterIdFlag' that, updated method 'generateMetadata' to set requester Id flag in EidasMetadataParametersI for idpRole)

EIDASValues.java (added REQUESTER_ID_FLAG)

Test code:

EidasNodeMetadataGeneratorTest.java (added 3 tests for Requester Id flag)

eidasKeyStore_Service_CA.jks (new file)

EncryptionConf.xml (new file)

EncryptModule_Service.xml (new file)

SamlEngine.xml (updated "Metadata" engine configuration)

Configuration changes

The following configuration files were added/changed/updated:

/default/eidas.xml

Was added the new property to enable/disable the publishing of Requester Id flag

<entry key="requester.id.flag">false</entry>

It was set to false in the default/eidas.xml which disables it publishing. If to be enabled, it should be set on external configuration eidas.xml in the following manner

<entry key="requester.id.flag">true</entry>

17. RequesterID is forwarded from Connector to ProxyService only when SPType is private

The outcome of the MS's consultation page was to forward the RequesterID into eIDAS SAML Request to ProxyService only when SPType is private and the RequesterId flag of Proxy-Service is set, therefore the code was adapted.

Code changes

AUCONNECTOR class:

The SpType of LightRequest and Connector metadata are checked. When the value is private the RequesterID will be added to the eIDAS SAML Request if the ProxyService has requester.id.flag set in its Metadata.

AUCONNECTORTestCase class:

9 new test methods along with auxiliary methods:

'testGetAuthenticationRequestWithPrivateSpTypes',

'testGetAuthenticationRequestWithPublicSpTypes',

'testGetAuthenticationRequestWithPublicRequestSpType',

'testGetAuthenticationRequestWithPrivateRequestSpType',

'testGetAuthenticationRequestWithPublicConfigSpType',

'testGetAuthenticationRequestWithPrivateConfigSpType',

'testGetAuthenticationRequestWithNullSpTypes',

'testGetAuthenticationRequestWithPrivateConfigPublicRequestSpTypes',

'testGetAuthenticationRequestWithPublicConfigPrivateRequestSpTypes'

EidasNodeMetadataGeneratorTest class:

2 new test methods along with auxiliary methods:

'testGenerateConnectorMetadataWithPrivateSpType',

'testGenerateConnectorMetadataWithPublicSpType'

18. Remove validation of ProviderName as not mandatory

While implementing the requesterId functionality to align with the technical specifications 1.2, it has been noticed that the presence of the ProviderName was validated to be mandatory.

There is no requirement for this neither in the eIDAS technical specifications nor in SAML core.

Therefore, the EidasProtocolProcessor class, the EidasAuthnRequestValidator and the AUCONNECTORSAML class were modified to remove the validation of the ProviderName which was only validating that a value was present.

Other files were also modified to remove the error messages that are not used anymore due to the removal of the validation. The files are:

- EidasErrorKey.java
- eidasErrors.properties
- errors.properties
- sysadmin.properties

© Copyright European Commission —

19. Replace Glassfish 5 full platform support by Glassfish 5.0.1 Web profile

With the update of the signature algorithm whitelist, it was noticed that the node encountered some issues when signing with some of the algorithms on the Glassfish 5 full platform.

Therefore, we decided to move our support for the Glassfish 5.0.1 web profile, which works without issues.

20. Adaptation of the eidas.protocol.version from 1.1 to 1.2

In the context of the implementation of the technical specifications 1.2, the configuration of eidas.xml need to be updated

Configuration changes

The value of the eidas.protocol.version parameter has been changed from the value 1.1 to 1.2 in the eidas.xml of the EIDAS-Config/server folder

21. Publishing custom list of eIDAS attributes at ServiceMetadata

In order to make it easier for the Member states to configure a custom list of published eIDAS attributes in the ProxyService metadata some modifications were made to the Node.

Code changes

The AUSERVICEUtil class was updated to be able to filter the list of supported attributes without unsupported attributes.

The AUSERVICESAML class was modified to add some validation on the requested attributes list to check if the ProxyService can correctly handle the request based on its actual unsupported attributes.

The EidasNodeMetadataGenerator has also been adapted to only publish the supported attributes to the ProxyService metadata (meaning that the unsupported attributes would be filtered out from the protocol engine configured attributes before being published in the metadata).

Therefore, the direct use of node properties (nodeProps) in the EidasNodeMetadataGenerator has been deprecated to the use of an AUNODEUtil instance. The applicationContext.xml has been modified to suit with this new behaviour.

The EidasParameterKeys class was changed to integrate the new "unsupported.attributes" parameter.

Configuration changes

A new eidas parameter "unsupported.attributes" has been added and should be filled with a comma-separated value of NameURIs of the attributes that the ProxyService side

© Copyright European Commission —

will not able to handle. If this parameter is not defined or is empty, the Node will just consider that the Node can handle all the attributes configured in the protocol engine.

22. Extension of the LightResponse with Consent and setting the consent in the eIDAS SAML Response

In order to fix the incorrect value of the consent previously used in the eIDAS SAML response, the LightResponse has been extended with a new "consent" field.

This new field should be filled with the consent enum values from the saml core specifications and the consent will be copied from the LightResponse to the eIDAS SAML response. For interoperability, the consent field is optional and if not filled in the default "*urn:oasis:names:tc:SAML:2.0:consent:unspecified*" value will be used.

Code changes

The LightResponse.xsd file has been updated in the source.

The ILightResponse interface, the AbstractLightResponse and the AbstractAuthenticationResponse classes have been updated as well to integrate the new consent field.

LightResponseTest class has also been updated for this new field.

The marshallResponse method from the EidasProtocolProcessor was updated to set the received consent value in the eIDAS SAMLResponse

A SAMLConsent enumeration class was created to represent the SAML consent possible values

An util method was made to get the consent value in the SAMLEngineUtils class

Unit tests, to check if the consent is correctly set in the SAMLResponse, were added in the EidasSAMLResponseSyntaxTest class

23. Extension of the LightRequest with spCountryCode and removing the option to forward spCountryCode via citizenCountryCode

In order to correctly forward the SP country code from proxy service to specific proxy service, the LightRequest interface and implementing abstract classes were extended to include SP country code

Code changes

The following files were changed:

- AbstractLightRequest.java: added spCountryCode to the attributes, updated the builder to accommodate the new attribute and implemented 'getSpCountryCode' method
- AbstractAuthenticationRequest.java: updated light request builder to accommodate the new attribute and implemented the 'getSpCountryCode' method
- ILightRequest.java: added 'getSpCountryCode' to methods
- LightRequestTest.java: added spCountryCode to builder tests

- ColleagueRequestServlet.java: removed 'updateCitizenCountryCodeValue' method and calls to this method.
- PropertiesUtil.java: removed isReplaceCitizenCountryCodeBySpCountryCode boolean
- EidasParameterKeys.java: removed key REPLACE_CITIZEN_COUNTRY_CODE_BY_SP_COUNTRY_CODE_IN_PROXYSERVICE_LIGHT_REQUEST
- ColleagueRequestServletTest.java: disabled tests relying on removed parameter key
- PropertiesUtilTest.java: removed tests for method 'updateCitizenCountryCodeValue' in class ColleagueRequestServlet.java

Configuration changes

- eidas.xml: removed configuration key:replace.citizenCountryCode.by.spCountryCode.in.proxyService.lightRequest

24. Level of Assurance Validation for Eidas Request and Eidas Response

In order to validate non notified levels of assurance, the current validation was moved into separate classes for request and response respectively.

Code changes

The following files were changed:

- LevelOfAssuranceRequestValidator.java
  New class to validate the incoming and outgoing levels of assurance in the Request.

- EidasRequestedAuthContextValidator.java
  New class to validate incoming comparison and levels of assurance

- LevelOfAssuranceResponseValidator.java
  New class to validate the incoming and outgoing levels of assurance in the Response.

- LevelOfAssuranceUtils.java
  Util class to help filtering level of assurance lists for notified and/or non-notified levels of assurance.

- EidasProtocolProcessor.java
  Added validation to the processing of the authentication request and response.

- AbstractLevelOfAssurance.java
  Correction to the setting of the type attribute

- AbstractLightRequest.java

© Copyright European Commission —

getLevelOfAssurance() no longer returns first element but NotifiedLevelOfAssurance

AbstractLightRequest.Builder.levelsOfAssurance() now trims notified levels of assurance and keeps lowest entered value

- AbstractAuthenticationRequest.java
  getLevelsOfAssurance() now adds in any required higher loas compared to lightRequest

- LevelOfAssuranceRequestValidatorTest.java
  Test class added to cover the LevelOfAssuranceRequestValidator

- EidasRequestedAuthContextValidatorTest.java
  Added test class to cover EidasRequestedAuthContextValidator.java

- LevelOfAssuranceResponseValidatorTest.java
  Test class added to cover the LevelOfAssuranceResponseValidator

- AUCONNECTORSAMLTest.java
  Adjusted order of loas to account for new methods

  EidasRequestedAuthnCon

- LightRequestTest.java
- Adjusted test to account for new methods
- AbstractEidasAuthenticationRequestBuilderTest.java
  Adjusted test to account for new methods

25. Level of Assurance Type Validation for Light Request

In order to ensure the correct use of Levels of Assurance within light Request, a validation at the connector's incoming light request has been added.

This validation checks the xml, if the "levelOfAssurance" field value is correct according to its type, which can be notified or nonNotified.

Consequently the definition of LevelOfAssuranceType has been refined.

Code changes

IncomingLightRequestValidatorLoAComponent.java to validate the levelOfAssurance fields in light request. Test covering this class have been added.

LevelOfAssuranceType enum has been updated to restrict levels of assurance according to these rules:

- Notified levels of assurance must exact match the strings defined in technical specification 1.2 Document SAML Message Format. (As opposed to URI matching)
- Any other levels of assurance cannot use the EIDAS Prefix, no matter the capitalization used.

This will affect LoAs in LightRequest/Response and EIDAS Request/Response
Tests were added to LevelOfAssuranceTypeTest

26. Validation for published Levels of Assurance on the Proxy Service

A proxy service publishes a list of Levels of Assurance (LoAs) in its metadata page to express what it is capable of processing. Now that specification 1.2 defines support for multiple published LoAs, all supported notified LoAs have to be in this list. Furthermore, to provide support for specification 1.1, the first LoA in the list also needs to be the highest notified LoA the proxy service supports.

In order to enforce this, validation has been added in the proxy service for connectors that only implement specification 1.1, in order to not send any incorrect LoA. The connector will also perform this validation with any proxy service in order to keep the same behaviour as previous connectors. Result is a 000009 Error by the connector or proxy service.

Code changes

EidasNodeValidationUtil.java was updated with the utility method 'isFirstLoaIsHighestNotifiedLoa'.

AUCONNECTORSAML.java was updated implementing this method & tests where added.

AUSERVICESAML.java was updated implementing this method & tests where added.

27. Digest method algorithm configuration

In order to allow MS to configure the digest method algorithm to use and the whitelist of digest method algorithm they authorized, two new properties were added.

Code changes

The SignatureConfiguration class has been modified to also have a digest method algorithm and digest method algorithm whitelist.

The KeyStoreSignatureConfigurator has been updated to (in addition to what it already used) either use the configuration of the digest method algorithm and digest method algorithm whitelist from properties or if empty use the default configuration which will be the "http://www.w3.org/2001/04/xmlenc#sha512" and the whitelist will be the list of the following digest method algorithms:

- http://www.w3.org/2001/04/xmlenc#sha256
- http://www.w3.org/2001/04/xmldsig-more#sha384
- http://www.w3.org/2001/04/xmlenc#sha512

The AbstractProtocolSigner, EidasNodeMetadataGenerator were also updated to take into account the new possible configuration of the digest method algorithm and the digest method algorithm whitelist.

© Copyright European Commission —

Configuration changes

Two new properties can be added to the eidas.xml:

- digest.method.algorithm

- digest.method.algorithm.whitelist

28. Fix digest algorithm list in Node metadata and preference order of algorithms

Metadata was previously published with an invalid list of digest method algorithms.

Now, the metadata shows the correct list of digest method algorithms which are handled by the node.

Code changes

The reasonable preference order configuration is now also possible for the signature and encryption algorithm. The node will now take into account the order of the values of algorithms given in the eidas.xml for the properties:

- signature.algorithm.whitelist
- encryption.algorithm.whitelist

A validation of the digest method algorithm used (against the digest method whitelist) is now also performed when verifying a signature.

29. Validation of the response in representation cases

The update of the Technical specifications introduced the need to validate that response matching with a representation case of authentication have always two Minimum data sets.

Therefore, a method that verifies if it is in representation case and if the attributes of the response are only from 2 Minimum Data Sets was added in the EidasProtocolProcessor.

This new method is called from the AUSERVICE at the moment of the validation of the idp response.

Unit tests for the new validation method have also been added to the EidasProtocolProcessorTest class.

Configuration changes

The external configuration entry "disable.check.representative.attributes" has been removed from eidas.xml along with the support for it.

30. Remove unnecessary validation of connector destination url

The processing of the LightRequest in AUCONNECTORSAML class was validating the url at the ProxyService to which the request has to be send.

This validation was in a way duplicated since the url is already extracted from the trusted metadata of the ProxyService itself. (elements: <md:SingleSignOnService>)

Code changes

The enum values holding these keys have also been cleaned up in EIDASValues.java

Configuration changes

Configuration keys related to the validation has been removed from eidas.xml:

- connector.url.redirect.location.whitelist
- connector.url.post.location.whitelist

31. Connector validation for NameID format

A validation has been added to the connector for the NameID format attribute. The length of each attribute in eIDAS request and eIDAS response is checked, so that it fits in the maximum length of 256 characters (in the case of NameID with persistent or transient format) or of 1024 characters (in the case of NameID with entity format), and if the length is exceeded, the following error message is displayed.

If the NameID format from eIDAS request is not "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" and if the NameID format from eIDAS response does not match the NameID format from the eIDAS request, a lightResponse is returned with statusCode = "Responder" and subStatusCode = "InvalidNameIDPolicy".

Additionally, it was also made possible to configure other NameID formats in addition to the persistent, transient and unspecified NameID Formats.

Additional in the request, the unspecified NameIdPolicy Format will be added if none is specified in the light request. This only in older (1.1 Spec) proxy services for interoperability reasons.

Code changes

AUCONNECTORSAML class:

Method 'checkNameIdFormat' was added to check if the NameID format from eIDAS response matches with the NameID format from eIDAS request, and also 'validateNameIdFormatLength' method was added to check the length for the NameID format (each NameID format attribute from eIDAS request and eIDAS response).

'checkNameIdFormat' method is called within the 'processProxyServiceResponse' method, while ''validateNameIdFormatLength' method is called within the 'checkNameIdFormat' method.

© Copyright European Commission —

Method 'validateRequestNameID' was added to check if the NameID Format requested is supported by both the connector and the proxyService and is used in the 'processSpRequest' method. If the NameID Format requested is not supported and error message will be logged and an error screen showing that the LightRequest is invalid will be shown.

'getNameIDFormatSet' method was also added to be able to fetch the actual set of connector's supported NameID formats.

Tests were added in AUCONNECTORSAMLTEST class in order to cover this, while in AUCONNECTORSAMLTestCase class, NameID format attribute was added so the tests pass successfully.

Others classes were adapted to publish the correct list of supported NameIDFormat in the metadatas, here is the list of the modified classes:

- EidasNodeMetadataGenerator
- EidasMetadata
- EidasMetadataRoleParameters
- EidasMetadataRoleParametersI
- MetadataUtil

AUSERVICESAML class:

Method 'checkNameIDFormat' was added to verify that the requested NameID Format is supported by the ProxyService, if it is not supported than an SAML error message is return with a new error code (202022) and message.

Configuration changes

New configuration keys related to add support for additional NameID formats can be used in eidas.xml:

- connector.optional.nameid.formats
- service.optional.nameid.formats

32. Generation of LightRequest LightResponse Classes from XSD

Due to multiple incoherences between the XSD model for the LightRequest and the LightResponse, the code was changed to generate a LightRequest and a LightResponse class from the xsd model.

Therefore the org.codehaus.mojo:jaxb2-maven-plugin:jar:2.3.1 was added in the EIDAS-specific-communication-definition module, to generate the classes from the xsd.

Code changes

The LightJAXBCodec class has been modified to use two JAXBInstance using the new generated java classes for the marshalling and unmarshalling of the LightRequest and LightResponse.

A new class LightMessagesConverter has been created to map the ILightRequest and ILightResponse to the new model and the other way around too. This was done to allow the previous model to continue being used in the Node, which will avoid interoperability issues.

The NamespaceAware attribute for the SAXParserFactory has been set to true in SecurityUtils.

XSD files for LightRequest and LightResponse were added to the module resources.

Configuration changes

When producing an xml that is compliant with either schema, it is now mandatory to include the namespace in the root element as attribute: (xmlns="http://cef.eidas.eu/LightRequest" or xmlns="http://cef.eidas.eu/LightResponse").

33. Other fixes/improvements requiring no action

CVE-2018-10237

Version of guava was upgraded from 19.0 up to 24.1.1-jre to resolve the CVE-2018-10237 vulnerability.

CVE-2020-1963

The ignite version was updated from 2.7.0 to 2.8.1 to solve CVE-2020-1963 vulnerability.

CVE-2019-12400

Version of the org.apache.santuario:xmlsec dependency was upgraded, to 2.1.4 version, to remove the code vulnerability to the CVE-2019-12400.

CVE-2015-0254

The taglibs-standard-impl was replaced by jakarta.servlet.jsp.jstl. Due to CVE-2015-0254, the dependency on the library javax.servlet:jstl:1.1.2 had to be changed.

In this context, the javax.servlet:jstl:1.1.2 and the org.apache.taglibs:taglibs-standard-impl:1.2.5 library used by the Node were removed.

The org.glassfish.web:jakarta.servlet.jsp.jstl:1.2.6 has been added in replacement for both of the removed dependency since this new library is also implicitly requiring the jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api:1.2.4 library.

Upgrade of se.swedenconnect.opensaml:opensaml-security-ext from 1.0.5 to 1.0.7

The se.swedenconnect.opensaml:opensaml-security-ext dependency version was updated from 1.0.5 to the latest version 1.0.7.

Update eIDAS Metadata VERSION 2.5.0 Application Identifier

Changed the eidas.application.identifier value from CEF:eIDAS-ref:2.4.0 to CEF:eIDAS-ref:2.5.0 in the external configuration file eidas.xml.

Node country consume from metadata

In previous version of the Node, the CountryCode retrieved from the signature certificate subjectDN would be used for two different purposes like:

- to fill the Unique Identifier Prefix
- to check the citizen country code matches with the proxyservice country code from which the eidasResponse is received.

The CountryCode used for these two purposes is now fetched from the Node metadata when possible/available, otherwise the country code will still be fetched from the signature certificate's subject to allow compatibility with older version of the node.

The EidasProtrocolProcessor has been updated to first try to get the Node CountryCode from the metadata of the colleague node.

Support for multiple application identifier values in Metadata

In the context of the implementation of the technical specifications 1.2, it is required for the node to be able to publish multiple application identifier values.

Therefore, a new method was added in the EidasMetadataParametersI interface to get the list of application identifiers.

The EidasMetadataParameters implementation was modified to be able to handle multiple application identifiers.

The EidasMetadata class was modified to publish the multiple values of application identifiers in different AttributeValue elements and the MetadataUtil class was modified to handle consuming metadata with multiple application identifiers.

Added unit test to test publishing of multiple application identifiers

The Eidas-Node now supports a comma-separated list of application identifier values.

Support for multiple protocol versions AttributeValue in Metadata

In the context of the implementation of the technical specifications 1.2, it is required for the node to be able to publish multiple protocol versions values.

Therefore, a new method was added in the EidasMetadataParametersI interface to get the list of protocol versions.

The EidasMetadataParameters implementation was modified to be able to handle multiple protocol versions.

The EidasMetadata class was modified to publish the multiple values of eidas protocol versions in different AttributeValue elements and the MetadataUtil class was modified to handle consuming metadata with multiple eidas protocol versions.

Added unit test to test publishing of multiple protocol versions in the metadata.

The Eidas-Node now supports a comma-separated list of protocol versions values for the eidas.protocol.version parameter.

© Copyright European Commission —

Metadata retrieval read for active.module.connector or active.module.service are false

When active.module.connector or active.module.service property was set to false, metadata was read and the logging point of the message appeared in the logs. Two new methods had been added in PropertiesUtil.java called "getProxyServiceConfigParameter" and "getConnectorConfigParameter" in order to retrieve the config parameters from ProxyService and Connector modules through AUSERVICEUtil.java and AUCONNECTORUtil.java classes. Also, the "getConfigParameter" method code has been simplified. The active check for the metadata is performed on the first line of following files to prevent any code from running upon a HTTP request:

'SpecificConnectorRequestServlet.doGet()',

'ColleagueRequestServlet.doPost()',

'SpecificProxyServiceResponse.execute()',

'ColleagueResponseServlet.doPost()',

'ConnectorMetadataGeneratorServlet.doGet()',

'ProxyServiceMetadataGeneratorServlet.doGet()'.

Flow fails with Invalid Light Response error if the connector metadata is not reachable

When ProxyService could not reach Connector's metadata because of 'active.module.connector' property with false value, an error was thrown related to 'Invalid Light Response.'. In order to fix this, 'SAML_ENGINE_NO_METADATA' error is thrown on 'EidasProtocolProcessor.getEncryptionCertificate' method.

Extension of the LightRequest for NonNotified Level of Assurance

In the LightRequest.xsd document the LevelOfAssurance element was modified

- with maxOccurs becoming "unbound"
- with the removal of the restriction on "notified" LoAs value and
- with the addition of the new "type" attribute

The instance variable "String levelOfAssurance" in the LightRequest class has been modified to a list of ILevelOfAssurance

The ILevelOfAssurance is a new interface which has been introduced to represent the LevelOfAssurance xml object having a type and a value.

The ILevelOfAssurance is also regrouping constants representing the notified eIDAS LoA prefix and values.

The ILevelOfAssurance is implemented within the AbstractLevelOfAssurance class which is itself extended by the LevelOfAssurance class.

© Copyright European Commission —

A new LevelOfAssuranceType enum has been created and contains the two values "NOTIFIED" and "NON_NOTIFIED", that have to be used to represent the LevelOfAssurance type.

Tests have been added in the LightRequestTest class to verify that the builder of the object works correctly.

The addition of the list of ILevelOfAssurance in the LightRequest was requiring a modification in the JAXBContext used for the instantiation of the LightJAXBCodec.

Therefore, some modifications to the LightJAXBCodec class to provide a build method with the default JAXContext was created and the other classes that use the LightJAXBCodec have been modified to use this new build method.

Extension of the EidasAuthenticationRequest for Non-Notified Levels of Assurance

A new "nonNotifiedLevelsOfAssurance" field has been added to the AbstractEdiasAuthenticationRequest class in order to represent the list of non-notified levels of Assurance values that could be received in the LightRequest.

The IEidasAuthenticationRequest interface has been accordingly modified with a new method to get these "non-notified" levels of assurance.

A new method has been added to the ProtocolProcessorI interface to introduce a 'marshallRequest' method taking a IEidasAuthenticationRequest in parameter instead of only a IAuthenticationRequest. The EidasProtocolProcessor implementation has been modified to implement this new method and to handle the possible multiple levels of assurance received in the LightRequest.

The marshalled eidasRequest will have a RequestedAuthnContext containing all the levels of assurance (and even the higher notified levels of assurance, if a notified Level of Assurance is present) if at least one non notified level of assurance is present in the LightRequest, the comparison type of the RequestedAuthnContext will be "Exact". In the case that only one notified level of assurance is present in the LightRequest and no non-notified levels of assurance are present, the RequestedAuthnContext will have only one level of assurance (notified) and the comparison type will be set as "Minimum".

Forward of Non-notified Levels of Assurance from EIDAS Request to Light Request

In the context of the alignment of the code of eIDAS node with the Technical Specifications v1.2, it is now possible for Member States to utilise non-notified levels of assurance.

To forward these non-notified levels of assurance from the EIDAS Request to the Light Request, changes were made in EidasProtocolProcessor.java

Further changes were made in EidasSamlRequestSyntaxTest.java and in SyntaxTestUtil.java

© Copyright European Commission —

Update the validation of the Levels of Assurance

With the introduction of the non-notified Levels of Assurance, with the appliance to the technical specifications 1.2, the validation of the levels of assurance received, on the connector side, in the LightRequest and from EidasResponse had to be updated.

Therefore, the AUCONNECTORSAML and AUCONNECTORUtil classes have been modified. In the AUCONNECTORSAML, the 'processSPRequest' method has been changed to first update the RequestState correctly with the whole list of levels of assurance received and to correctly validate the list of Levels of Assurance against the proxy service metadata list of allowed/available Levels of Assurance. For more details please check Appendix A.eIDAS Levels of Assurance of eIDAS-Node Installation and Configuration Guide.

The 'checkResponseLoA' method has also been updated to validate the response level of assurance to check if it matches with the requested list of Levels of Assurance.

The 'validateSP' method of the AUCONNECTORUtil class has been modified to avoid the validation of the levels of assurance to only be notified Levels of Assurance.

Link to these modifications, some unit tests were added in the AUCONNECTORSAMLTest class.

Allow validation for non-notified LoA in proxyservice

Existing validation of Levels of Assurance in the proxyService was extended to allow for non-notified schemes in Level of Assurance.

Changes were made in EidasProtocolProcessor.java and in EidasProtocolProcessorTest.java

Validation of Levels of Assurance in Light Request Builder

Validation was added to the builder of LightRequest to ensure that a light request has at least one level of assurance. Furthermore, levels of assurance objects can now only be created from an absolute URI string.

Code changes

Changes were made to AbstractLightRequest.java and AbstractLevelOfAssurance.java. Furthermore, tests were added to LightRequestTest.java and LevelOfAssuranceTest.java. New utility method has been added to Preconditions.java to check if a string is a valid absolute URI.

Additionally MessageLoggerUtils and EidasNodeValidation have been updated accordingly. Also all tests using LightRequest or LevelOfAssurance have been updated:

- LightRequest tests that were not using LoA's in their lightRequest now have at least one LoA.
- Test that use LevelOfAssurance with a now non-compliant LoA string value have been aligned.

© Copyright European Commission —

Level of Assurance backwards compatibility with EIDAS protocol version 1.1

When sending an authentication request to a proxy service that identifies as using EIDAS protocol version 1.1, any non-notified levels of assurance will be removed from the request to provide backwards compatibility for notified levels of assurance.

The resulting request will contain one notified level of assurance and comparison minimum. If the original request did not contain any notified levels of assurance the connector will display an error message as sending an empty request is not possible.

Code changes

Changes were made to AUCONNECTORSAML.java and tests have been added to AUCONNECTORSAMLTests.java.

Response Level of Assurance Correlation with Service Metadata Levels of Assurance.

When receiving a response in the connector, the response is checked against the request from which it originated. This to determinate if the response Level of Assurance is good enough for the request. Here we have added an additional validation: The response LoA cannot be higher what is published as the highest LoA level in the metadata of the proxy service, if this is not the case the response will get rejected.

For a proxy service implementing the 1.2 specification this is even stricter: Only LoAs (multiple) published in the proxy service metadata can be valid in the response.

This means that if a proxy service responds with a LoA higher than stated in the metadata the response will get rejected.

Code changes

The comparing against the proxy service metadata was added in the method in AUCONNECTORSAML. A new method was added to EidasNodeValidationUtil.java to be more flexible at comparing LoAs. Tests where added for both classes, additionally any test that covered response was updated to also have metadata parameters.

Log the complete Light Response received by the Proxy-Service

The implementation has been updated in such a way that the lightResponse method should be the same as the Light Response from the cache. New "getResponse" method has been added to SpecificCommunicationLoggingService interface which returns lightResponse as a String. Also some code from AbstractSpecificCommunicationLoggingService has been extracted to 'getLightResponseFromTokenBase64' method which is called in 'getResponse:ILightResponse' and 'getResponse:String' methods.

© Copyright European Commission —

Log the complete Light Request received by the Connector

In order to log the light request received in the Connector, a new method that returns a string called 'getRequest' has been made on the SpecificCommunicationLoggingService interface. This gets implemented by SpecificConnector and SpecificProxy Communication Logger Services and consumed by ConnectorIncomming and ProxyserviceOutgoing Request Loggers.

Truncated messages logged in errors based on maximum message length

Added an extra validation when logging messages in error. When message exceeds maximum allowed message size, the logged message will be truncated so that it does not exceed the maximum allowed size in the logs.

Added test class NormalParameterValidatorTest to test correct validation of message lengths

Modification of logging processing

Previously, if an error occurred during the logging of an incoming request or response, the Node would not process the request/response. This behaviour has been changed since the behaviour of the Node shouldn't be different when the logging is active or inactive.

Therefore, loggers are now catching the exceptions that would occur during the process of the logging.

The following logger classes are impacted:

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

The "UNDEFINED" logging constants has also been introduced, so when an exception occurs during the logging, the loggers will still log all the information it could determine and print the rest with an value "UNDEFINED".

Unit tests to cover this have been added, and others have been updated.

SAML Response message is exposed in the logs when sending big light response

Saml Response is exposed in the log file when lightResponse is too big. This has been fixed, so when Saml Response length is too big, it will not be logged in the file.

Add validation for the key length of the certificates used to sign metadata and authentication request

In order to align with the technical specifications 1.2 and more precisely the cryptographic requirements, validations of the certificate key length were added during the signature validation.

The AbstractProtocolSigner was updated to include some key length validation in the 'verifyCryptographicSignature' method which is used by the public 'validateSignature' method of the same class.

The ProtocolSignerTest class and the signatureTestKeystore.jks were added to test the validation of key length.

Due to this modification, the metadataPlusExtraCertificate.jks was also adapted to not generate issues within the tests where it is used.

Add validation of the signature algorithm's hash length of the certificates used to sign the metadata and/or authentication request

In order to align with the technical specifications 1.2. and more precisely the cryptographic requirements the hash length of the signature of the certificate used to sign the metadata or request has to be equals or bigger than 256 bits.

Therefore, some modifications were done to the AbstractProtocolSigner class to add this validation to all the certificates used to sign.

Some tests were added to verify the behavior of the newly introduce methods:

The 'vaidateCertificateSignatureHash' method which takes a X509Certificate in parameter

And the 'getHashAlgorithmBitsLength' method which takes a hashAlgorithmName in parameter, and multiple keystores and tests configurations files were modified to not generate issue due to this added validation.

In the EIDAS-Config module, the following files:

- /keystore/eidasKeyStore_METADATA_TC.jks
- /server/SignModule_Connector.xml
- /server/SignModule_Service.xml

were updated with a new PrivateKey entry and trust entries compliant with the new validation.

Implementation to determinate digest algorithm

In some cases the digest algorithm was not determined in a correct way. Implementation in AbstractProtocolSigner was adjusted accordingly. Responsibility has been move to the SAMLEngineUtils class and tests were added for the new implementation.

© Copyright European Commission —

Update EUID attribute validation

Due to the technical specifications requirements, the AUSERVICE was modified to add a specific validation for the Unique Identifiers (UID) attributes values. The specific validation takes into account a possible identifier prefix.

A ValueValidator interface defining a method 'isValid' with a value parameter where the parameter type is defined by the implementation class.

Two concrete ValueValidator classes were added for:

- a string value which length is validated based on parameter name for which a max size configuration parameter can be found.
- a string value validated with a given pattern.

Together with these validator classes, an AttributeValidator class was created to retrieve a specific ValueValidator implementation based on an attribute definition.

The validation of the Attributes has also been added on the Connector where it was missing, in the 'processProxyServiceResponse' method of the AUCONNECTORSAML class.

Update value and validation of Gender possible values based on protocol version

Due to the implementation of the technical specifications 1.2, it is good to differentiate when values are valid for one or other versions of the specifications.

In the context of the gender attribute, a "new" value "Unspecified" was introduced to replace the "Not specified" value. Since the node should support both versions of the specifications (1.1 and 1.2), the validation of the values in the Gender Attribute needs to take into account the protocol version of the Node.

Therefore, some modifications were applied to the code of the Node to make this possible.

The EidasProtocolVersion enum class was added to represent the eIDAS protocol versions

A ProtocolVersionValidator abstract class was created to be a base for the GenderProtocolVersionValidator class which validates the gender based on one or multiple protocol versions.

A method was added in the AttributeUtil class to check if an attribute is representing a Gender or not.

A new method was added to the ProtocolProcessorI interface to get the metadata protocol version from a metadata url. This new method was implemented in the EidasProtocolProcessor class.

The AUSERVICE class was updated to modify the Gender value to align with the destination connector's supported protocol versions

The AUCONNECTORSAML class was updated to validate the Gender value based on its (the connector) supported protocol versions

© Copyright European Commission —

Multiple unit tests were added or updated according to these changes in the following classes:

- AttributeUtilTest
- AUCONNECTORSAMLTest
- AUSERVICETestCase
- EidasProtocolProcessorTest
- GenderProtocolVersionValidatorTest

Citizen country code validation

Due to the introduction of the NodeCountry element in the Node metadata, the validation of the citizen country code with the service country code had to be changed.

Therefore, if the service country code is fetched from the Proxyservice's metadata and not from the signature's certificate, the validation of the country code is always executed.

But if the country code is still retrieved from the signature's certificate, than the validation only occurs if the property "*check.citizenCertificate.serviceCertificate*" is set to true.

The code of 'checkServiceCountryWithCitizenCountry' method in the AUCONNECTORSAML class has been updated with the above mentioned adaptation.

A new method in IProtocolProcessor interface has been added to expose the possibility to more specifically get the metadata NodeCountry code from a given metadata url, this method is implemented in the EidasProtocolProcessor class.

New tests to validate this new implementations have been added in the AUCONNECTORSAMLTestCase class.

Connector check the nationality code of a sending country in assertions

A validation was added to check that the 'from' country code in identifiers contained in assertions matches the country code of the sending proxy service. This validation will be disabled together with the prefix formatting validation by setting the "validate.prefix.country.code.identifiers" configuration to false.

Code changes

AUCONNECTORSAML class:

A new method 'checkIdentifierCountryCodeMatchesServiceCountryCode' was added, which checks that the 'from' country code in the received identifiers matches the country code of the sending proxy service.

This method is called within the 'processProxyServiceResponse' method.

Tests for this new method were added in AUCONNECTORSAMLTEST.

Additionally we added JavaDocs to 'loadConfigServiceMetadataURL' and deprecated 'loadConfigServiceURL' in AUCONNECTORUtil.java value element of Requested attributes

© Copyright European Commission —

in the LightRequestIn the LightRequest xsd, the requested_attributes were supposed to have a value element, this shouldn't have been. Therefore the value element has now been made optional.

Update VersionMismatch as StatusCode instead of SubStatusCode in LightResponse.xsd

In order to align the LightResponse xsd with the way the "urn:oasis:names:tc:SAML:2.0:status:VersionMismatch" code is used with the eIDAS Node and also with SAML Core, the code was removed from the enumeration values of the subStatusCode values and added to the statusCode values.

Validation of StatusCode and Substatus code values

To ensure xsd compliance of the LightResponse,

The EIDASSubStatusCode enum class and the EIDASStatusCode enum class were both updated to be aligned with the XSD restrictions and the IncomingLightResponseValidator class was modified to also validate the statusCode value and subStatusCode value of the response Status.

XML for failed LightResponse is not valid according XSD

An issue was raised that elements 'subject', 'subjectNameIdFormat' and 'levelOfAssurance' in the LightResponse XSD had a minOccurs value of 1, even though they are optional in the LightResponse. To remedy this, the minOccurs attribute of these elements was set to 0 in the LightResponse XSD.

Externalised maximum parameter length for EIDAS response/request validation

The maximum size limit of the EIDAS request and EIDAS response has been changed from a hardcoded value to the value set in the eidasParameters properties file, using the keys max.SAMLRequest.size and max.SAMLResponse.size respectively.

Discontinue support for sys property "org.opensaml.httpclient.https.disableHostnameVerification"

It was decided that is no specific need to keep support for sys property "org.opensaml.httpclient.https.disableHostnameVerification", therefore it was removed from OpenSamlHelper.java class. Also, BaseMetadataFetcherTest test class has been created with proper unit test in order to cover this. Details of this issue were established in EID-1046 (EIDINT-4340)

Remove config parameter "include.assertion.fail.response.application.identifiers"

The temporary configuration parameter "include.assertion.fail.response.application.identifiers" has been removed along with the code dedicated to this configuration. Also, tests have been added in order to cover this.

© Copyright European Commission —

Fix ProxyService Metadata generator usage of Metadata enabled flag

In the PropertiesUtil class, some methods like 'isMetadataEnabled' are used by both the Connector and ProxyService side of the Node. Problem was that these methods are using application parameters which can be retrieve through the AUCONNECTORUtil and AUSERVICEUtil class which are specific for the instance type of the Node.

Therefore this method 'isMetadataEnabled' has been deprecated and is replaced by the 'isConnectorMetadataEnabled' and 'isProxyServiceMetadataEnabled' and the usage of the deprecated method has also been adapted in the Node code.

So the ConnectorMetadataGeneratorServlet, the ProxyServiceMetadataGeneratorServlet and the AUCONNECTORSAML have adapted to use these new methods.

New unit tests have been created to cover the updated code in:

- ConnectorMetadataGeneratorServlet (new tests created in the new ConnectorMetadataGeneratorServletTest class)
- ProxyServiceMetadataGeneratorServlet (new tests created in the new ProxyServiceMetadataGeneratorServletTest class)
- PropertiesUtil (tests in the PropertiesUtilTest class)

The ColleagueRequestSevletTest class was also adapted related to this change

Removed Hazelcast from EIDAS-Commons

Removed EIDAS-Commons direct dependency on hazelcast, hazelcast-wm

Code changes

Classes HazelcastInstanceInitializer and ConcurrentMapServiceDistributedImpl were removed from eidas-commons. These classes still exist in the EIDAS-JCache-Hazelcast module.

Hazelcast and hazelcast-wm definitions and versions have been removed from EIDAS-Parent's pom.xml: Hazelcast is now a responsibility of the EIDAS-JCache-Hazelcast module.

Update SimpleProtocol version in EIDAS-Parent dependency management

The version of the SimpleProtocol dependency in the dependency management of the EIDAS-Parent pom has been updated to use the newer version (0.0.3) of the SimpleProtocol.

Commented code removed from Application Context xml files

Commented code dealing with a resolved issue was removed from following files:

- specificConnectorApplicationContext.xml
- specificProxyServiceApplicationContext.xml

© Copyright European Commission —

Removed initialization of SubStatusCode in ResponseStatus factory method

Since the subStatusCode present in the Status of the LightResponse is not a mandatory element of the XML, the initialization of this element to the value "##" was removed from the newInstance factory method called during the unmarshalling of a LightResponse.

Introduction of MessageLog model to replace StringBuilders in Loggers

Code changes

Introduction of the package eu.eidas.node.logging.messages and of the following classes:

- MessageLog
- LightRequestMessageLog
- LightResponseMessageLog
- EidasRequestMessageLog
- EidasResponseMessageLog

MessageLog is an abstract class representing several tags that will commonly been output for each type of message received by the Node.

In order to represent the tags, the MessageLog class introduce a static inner class Tag. This class defines an output format for each Tag which consists of a title padded left on 14 characters followed by the tag value.

The MessageLog class also defines a static abstract Builder class, which provide setters for all the common tags and a generic build method.

The other classes are an extension of the MessageLog providing additional tags specific to the Message type and a builder extending the MessageLog.Builder.

The following connector and proxy service logger classes have also been adapted.

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

The StringBuilders previously used to create the MessageLog have been replaced by the use of the newly introduced MessageLog Builders in accordance with the type of MessageLog to output.

Unit tests are also added to cover the new message log representation.

Fix incorrect constant value (cfr. (EID-636) Question on "EIDAS SAML Engine 1.4.1")

In EidasSpec.java, the "EidasSpec.REPV_LEGAL_PERSON_IDENTIFIER" was pointing to LegalPersonSpec.Definitions.LEGAL_PERSON_IDENTIFIER instead of RepresentativeLegalPersonSpec.Definitions.LEGAL_PERSON_IDENTIFIER.

© Copyright European Commission —

This has been fixed.

Fix duplicate filtering of Outgoing EidasResponse in ProxyService

In case of outgoing error response from the ProxyService, a duplicate filtering of the outgoing response happened causing a duplicate log of the EidasResponse.

This was solved by removing the "/InternalExceptionHandler" from the url patterns of the ProxyServiceOutgoingEidasResponseLoggerFilter WebFilter.

Fixed concurrency issue in LoggingConcurrencyTest

Arraylist was being used by multiple threads to fill up test log file, which is not thread-safe. Enclosed the Arraylist in an instance of Collections.synchronizedList which should make this thread-safe, preventing test failures.

Javadoc improvement: Reduce Javadoc warnings

There were 34 warnings during generation of javadoc. In order to fix that, javadoc was updated and this has been resolved.

(EID-1051) Fix incorrect usage of retrieveConnectorAttributes

In the ProxyServiceOutgoingLightRequestLogger and the ProxyServiceIncomingLightResponseLogger classes, the 'retrieveConnectorAttributes' method from the MessageLoggerUtils class was used while the 'retrieveProxyServiceAttributes' method should have been used since the code is used in the ProxyService.

This has been fixed.

Creation of an default eidas.xml configuration file

In order to clean up the "external" eidas.xml configuration file, a new property file is introduced. This new file, called "eidas.xml", has been directly added in a new "default" folder of the EIDAS-Node resources.

The values of these properties will be overridden by the value specified in the external eidas.xml configuration file.

Avoid Ignite to make http request to verify if version is up-to-date

In order to avoid HttpRequests from the Node to Internet when Ignite caches are used, a IgniteUtils class was added to the EIDAS-JCache-Ignite module.

A static method 'deactivateIgniteVersionNotifierByDefault' was created and is used by the IgniteInstanceInitializerNode class to deactivate the default behaviour of Ignite.

The behaviour can still be activated by setting the IGNITE_UPDATE_NOTIFIER system property to true.

© Copyright European Commission —

Transform EIDAS-Config properties into a deploy-able set of property files

In Eidas-Config module, some files were changed in order to allow these files to be deploy-able without change for a domain localhost and a server using port 8080 and with http. The following lines describe the changes:

eidas.xml: The eidasnode:8888 was replaced by localhost:8080, The proxy-service countries configurations were updated. The properties based were regrouped based on relationship with one another. The formatting was also improved.

Inconsistencies in Eidas-attributes.xml config files

There were inconsistencies between eidas-attributes.xml file from SP, SpecificConnector, SpecificProxyService and IdP modules. In order to fix this, all eidas-attributes.xml files were updated to be consistent. Also, in AbstractPostalAddressAttributeValueMarshaller class, "unmarshallAddressWithoutDecode" method along with secondary methods, were added to process the address value without decoding it.

Fix non deterministic build failure build of tests in EIDAS-SAMLEngine Module

A non-deterministic build failure in the tests, in EIDAS-SAMLEngine module, was identified. The ResponseUtil class and ProtocolEngineTest class were updated to ensure the tests to succeed independently from the order of the tests execution.

Javadoc is pointing to itself

Updated methods from unit tests pointing to targeted class.

Unit tests performance improvements

In the context of the update of TLS protocol and ciphers suites regarding to the specifications 1.2., improvements of the performance were made in the HTTPSConnectionWithEIDASCipherSuiteTest class.

Remove rsa-ripemd160 from test code

All references to ripemd60 were removed from test code, several tests files were updated with the most recent metadata files, which have only the eIDAS 1.2 specification algorithms. In the course of this implementation the code was improved, in TestEidasNodeMetadataLoader, e.g. test method testValidatesignature was renamed to camelCase testValidateSignature, constants such as "METADATA" were extracted to fields, and unused code was removed.

Unit test failure on UNIX Cloud environment

'AUCONNECTORSAMLTest.processSpRequestWithActiveConnectorModule' test failed due regression. In order to fix this, SessionHolder.clear() was added in the setUp() method to make sure the ThreadLocal of the SessionHolder is empty before the test execution.

© Copyright European Commission —

Sonar: EIDAS-SpecificCommunicationDefinition: new vulnerability reported (minor)

A vulnerability was reported in the code of EIDAS-SpecificCommunicationDefinition module by Sonar. In order to fix this, "LIGHT_CODEC_CLASSES" access modifier was changed from public to protected.

CertificateUtilTest class refactoring and improvements

The class CertificateUtilTest was refactored and a new test method was added 'checkChainTrustCheckSelfSignedSigningCredentialTrustingSelfSignedSigningCredential' to test the case when the credential to be checked for trust is a self-signed one and the trusted credential is the same one. This is different from the other tests that use non self-signed credentials e.g. 'checkChainTrustCheckMetadataSigningCredentialTrustingMetadataSigningCredential'.

SAMLAuthnResponseDecrypterTest improvements

A 'tearDownClass()' method was added to the class SAMLAuthnResponseDecrypterTest to prevent interferences with other tests e.g. in CertificateUtilTest.

Regression: 'testConfiguredCacheMultipleExpiringKeys' unit test failure

IgniteCreatedExpirePolicyTest.testConfiguredCacheMultipleExpiringKeys test failed due to regression. In order to fix this, the cache expiry duration has been increased to 2 seconds and the number of keys has been decreased in order to execute the code no matter of the machine's performance.

Validation of LightRequest received by the Connector

There are multiple reason for a LightRequest to be invalid or unfit to unmarshalling therefore, a new error code (000011) and message have been added to be displayed when the LightRequest treated by the Connector is not valid.

For this new error code (000011) and message, the following files have been modified:

- EidasErrorKey.java in EIDAS-Common module
- eidasErrors.properties in EIDAS-Common module
- errors.properties in EIDAS-Node module
- sysadmin.properties in EIDAS-Node module

And the SpecificConnectorRequestServlet has also been adapted to handle error of the type and use the new error message and code.

(EID-1139) Invalid omitXmlDeclaration

In method 'marshall' of DocumentBuilderFactoryUtil the setting for the value omitXmlDeclaration was using "false" as one of the two possibilities which, is not one of the allowed values: "yes", "no". Therefore, the "false" was changed to "no".

© Copyright European Commission —

Removed decodeCurrentAddress javascript function

In method base64.js file, the function 'decodeCurrentAddress' was removed since it is not used anymore.

(EID-1136) Typos “metdata” instead of “metadata”

Various instances existed where “metadata” was misspelled as “metdata”

To fix this the following files were modified:

- EidasProtocolProcessorTest in EIDAS-SAMLEngine
- SamlEngine.xml in EIDAS-SAMLEngine

(EID-1115) Wrong error message in XmlSchemaUtilTest

XmlSchemaUtilTest.java was updated to allow an error message change made in JDK8u261 and later.

(EID-1135) - Typos "fecther" instead of "fetcher"

Various instances existed where “fetcher” was misspelled as “fecther” in test code files.

To fix this the following test files were modified:

- EidasProtocolProcessorTest in EIDAS-SAMLEngine
- TestMetadataFetcherThrowsException in EIDAS-SAMLEngine
- MetadataFetcher_ServiceMetadataFetcherThrowsException.properties in EIDAS-SAMLEngine
- SamlEngine.xml in EIDAS-SAMLEngine

(EID-1119) – eIDAS-Node Error Codes has not deprecated codes

Some error codes were marked as deprecated while still in use. Conversely, some appeared to still be in use while being deprecated.

Errors wrongfully marked as deprecated:

- 201005

Errors wrongfully not marked as deprecated:

- 203003

To fix this, the following files were updated:

- eidasErrors.properties
- errors.properties

- Sysadmin.properties

Upgrade org.apache.httpcomponents:httpclient to version 4.5.13

The dependency org.apache.httpcomponents:httpclient was upgraded from version 4.5.5 to version 4.5.13 to apply the recommended mitigation to address the CVE-2020-13956 (EID-1142).

© Copyright European Commission —