

eIDAS-Node Migration Guide v3.1.0

eID Internal

Table of Contents

1	Introduction	4
1.1	Document structure	4
1.2	Document aims	4
2	Prerequisites	5
3	eIDAS Node 3.1.0 Usage Statistics, Custom TLS Truststore and Extended Metadata Contacts	6
4	Changes	7
4.1	Summary of Implemented Stakeholder Issues.....	7
4.2	Keystore for TLS certificates	7
4.2.1	Custom keystore for TLS certificates	7
4.3	Usage statistics	8
4.3.1	Status-board Demo	8
4.3.2	Code changes	9
4.3.3	Configuration changes	11
4.4	More contacts	12
4.4.1	Code changes	12
4.4.2	Configuration changes	13
4.5	Update expired and almost expired certificates	14
4.5.1	Code changes	14
4.5.2	Configuration changes	14
4.6	Upgrade tomcat version to latest 11 patch version	14
4.6.1	Code changes	15
4.7	Other changes requiring no action	15
4.7.1	Upgrade apache commons-lang to commons-lang3.....	15
4.7.2	Upgrade logback-classic dependency version	15
4.7.3	Upgrade spring to 6.2.13.....	15
4.7.4	Unable to override IGNITE_MARSHALLER_WHITELIST due to improper default mechanism (EIDSD-1020, EIDSD-1007)	15
4.7.5	Set CacheManagers to Lazy	16

Document history

Version	Date	Modification reason	Modified by
1.0	29/05/2026	Created	DIGIT

Disclaimer

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

The document contains information of a technical nature and does not supplement or amend the terms and conditions of any procurement procedure; therefore, no compensation claim can be based on the contents of this document.

© European Union, 2023

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.

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 v3.0.0 to eIDAS-Node v3.1.0

1.1 Document structure

This document is divided into the following sections:

Chapter 1 — *Introduction*: this section.

Chapter 2 — *Prerequisites*: Identifies any prerequisites that are required before migrating your eIDAS-Node to version 3.1.0.

Chapter 3 — *Release Highlights*Chapter 4 — *Changes*: Contains detailed information about the changes that should be taken into consideration when migrating to eIDAS-Node version 3.1.0.

1.2 Document aims

The main aim of this document is to provide information on all the changes requiring your action when migrating to eIDAS-Node version 3.1.0, 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://interoperable-europe.ec.europa.eu/collection/eupl/eupl-text-eupl-12>

2 Prerequisites

Before starting your migration to eIDAS-Node version 3.1.0 you should have:

- Already implemented eIDAS-Node version 3.1.0
- Downloaded the eIDAS-Node v3.1.0 Integration Package; and
- Downloaded the latest documentation.

3 eIDAS Node 3.1.0 Usage Statistics, Custom TLS Truststore and Extended Metadata Contacts

This release introduces usage statistics support in the eIDAS-Node, including authentication flow tracking, long-term storage of flow data, and demo-oriented visualization and reporting capabilities through CSV export and InfluxDB-backed Status Board integration. The eIDAS-Node now also supports an optional application-scoped TLS truststore that adds custom trusted certificates for metadata retrieval while keeping the JVM default truststore available. In addition, the release introduces support for configuring and validating multiple metadata contacts through a dedicated XML configuration file, including additional contact types and XSD-based validation, while maintaining backward compatibility with the legacy eidas.xml contact properties.

4 Changes

4.1 Summary of Implemented Stakeholder Issues

EIDSD-1020, EIDSD-1007: Unable to override IGNITE_MARSHALLER_WHITELIST due to improper default mechanism

EIDSD-1012 eIDAS eID and known vulnerable dependencies

EIDSD-1041 eIDAS-Node Pre-Release 3.1.0 - LU comments

4.2 Keystore for TLS certificates

4.2.1 Custom keystore for TLS certificates

The eIDAS-Node now supports an additional, optional keystore dedicated to TLS certificates, defined with the purpose TLSTRUSTSTORE. TLSTRUSTSTORE adds to the JVM truststore, if no TLS certificates are configured, the JVM's default cacerts is used. This keystore allows operators to add custom trusted certificates for TLS validation without modifying the default JVM truststore. The BaseMetadataFetcher was updated to retrieve these trusted certificates via the signature configuration and to include them when creating the TLS socket factory. The newSslSocketFactory() method now uses these certificates to extend the set of trusted anchors for client-to-server (one-way) TLS connections during metadata retrieval.

Note: Using a custom keystore allows eIDAS-Node to trust certificates that may not be trusted by the host JVM. Operators should only add certificates they explicitly trust, as this can override system distrust decisions.

4.2.1.1 Code changes

- EIDAS-Node-Connector
 - updated connectorMetadataFetcher bean to inject the tlsTrustedCertificates property using the new tlsTrustAnchors bean in applicationContext.xml
- EIDAS-Node-Proxy
 - updated proxyServiceMetadataFetcher bean to inject the tlsTrustedCertificates property using the new tlsTrustAnchors bean in applicationContext.xml
- EIDAS-SAMLEngine
 - added TLSTRUSTSTORE to KeyStoreContent.KeystorePurpose to support a dedicated keystore for TLS trust anchors
 - updated getSignatureConfiguration method logic to load the optional TLSTRUSTSTORE subset and extract its certificates for TLS trust configuration in KeyStoreSignatureConfigurator.java
 - updated the subset method to include a mandatory flag and added a new constructor to support optional TLSTRUSTSTORE configurations without throwing errors in ListKeystoreContent.java
 - added tlsTrustedCertificates field, getter, and builder method for holding TLS truststore certificates in SignatureConfiguration.java
 - added new field tlsTrustedCertificates, initialized from SignatureConfiguration object in AbstractProtocolSigner.java
 - added new method getTlsTrustedCertificates() for safe retrieval of TLS certificates in AbstractProtocolEngine.java
 - added the getTlsTrustedCertificates() method to expose TLS trust certificates in ProtocolEngine.java

- EIDAS-Metadata
 - added `getCustomTrustManagers()` and updated `newSslSocketFactory()` to include custom trust anchors in `BaseMetadataFetcher.java`
 - added `CompositeX509TrustManager` class that extends `X509ExtendedTrustManager` and delegates validation across multiple trust managers, enabling combined use of JVM and custom truststores.
 - updated `BaseMetadataFetcherTest.java` with unit tests to verify the new behaviour with custom, empty and invalid truststores

4.2.1.2 Configuration changes

- EIDAS-Config
 - `server/connector/keystore`
 - added `eidasTlsTrustStore.p12` keystore
 - added `eidasTlsKeystore.p12` keystore
 - `server/connector`
 - added a new `TLSTRUSTSTORE` configuration entry for the custom TLS truststore used in secure metadata retrieval, in `SignModule_Connector.xml` and `SignModule_Connector_EC.xml`
 - `server/proxy/keystore`
 - added `eidasTlsTrustStore.p12` keystore
 - added `eidasTlsKeystore.p12` keystore
 - `server/proxy`
 - added a new `TLSTRUSTSTORE` configuration entry for the custom TLS truststore used in secure metadata retrieval, in `SignModule_Service.xml` and `SignModule_Service_EC.xml`

4.3 Usage statistics

This epic adds the base for collecting usage statistics inside the eIDAS-Node. The goal is to track how authentication flows move through the system, from the moment a request arrives until the response is sent back, and to store key information about their performance and result. A new object called `AuthenticationExchangeFlow` was created to hold this information. It contains details such as when the request and response started, how long they took, the origin or destination url, the level of assurance (LoA), and the final status of the flow. These flow details are kept in memory for a short time, so the node can build a clear picture of completed and ongoing authentication exchanges.

The eIDAS Node includes a worker to assess the final status of a flow when the time to complete the flow has expired. This worker will then write the result to long term storage. This will later make it possible to show meaningful usage statistics to operators, such as success rates or performance indicators, without having to manually analyze log files. These type of statistics can be grouped by status, connection and request level of assurance.

4.3.1 Status-board Demo

The eIDAS Node currently ships with 2 long term storage solutions for statistics, one of which is the InfluxDB time series database. We have create a demo application called the status-board that is connected to influx and uses the stored data to demonstrate how a rapport from this data can be visualized and downloaded.

The rapping currently stands outside the node and is considered a demo-oriented tool. The eIDAS Node functionality is limited to registering and sending the data used here. Exporting the

usage statistics to an InfluxDB OSS v2 database must be enabled before it can be used with the status board.

To configure the InfluxDB OSS v2 database, operators must provide an external InfluxDB OSS v2 instance for long term storage of usage statistics. The recommended approach is to install InfluxDB OSS v2 following the official documentation at <https://docs.influxdata.com/influxdb/v2/>, where both installation and initial setup (including user, organization and bucket creation) are described in detail.

For the eIDAS Node which consists of a connector and proxy service, a bucket must be created for each deployed application to store AuthenticationExchangeFlow records used for usage statistics. Each operator must generate an appropriate API token with write access to these buckets in their own environment and configure this token in eidas.xml. The node now pushes AuthenticationExchangeFlow data into the dedicated InfluxDB buckets (status-connector and status-proxy) for long-term storage and later retrieval by the Status Board, and this writing is asynchronous so it does not block the authentication flow.

Note: Usage statistics rely on the message logging functionality, so the saml.audit property must be set to true for statistics to be collected and exported.

For more detailed information on how usage statistics are logged, see section 7.4 *Usage Statistics Logging* in the *eIDAS-Node eID Error and Event Logging* document.

4.3.2 Code changes

- EIDAS-Config
 - server/connector
 - updated Hazelcast xsd configuration to hazelcast-config-5.6.xsd in hazelcastNode.xml and hazelcastSpecificCommunication.xml
 - updated cluster name from devServer to devServerConnector in hazelcastNode.xml
 - server/proxy
 - updated Hazelcast xsd configuration to hazelcast-config-5.6.xsd in hazelcastNode.xml and hazelcastSpecificCommunication.xml
 - updated cluster name from devServer to devServerProxy in hazelcastNode.xml
- EIDAS-Commons
 - removed several string constants no longer used after introducing MessageOperation enum in EidasValues.java
- EIDAS-JCache-Dev
 - updated iterator logic to return a functional Cache.Entry iterator over the internal ConcurrentMap to support flow scanning with JCache implementations in JCacheConcurrentMapAdapter.java
 - updated ConcurrentMapJcacheServiceDefaultImpl to use expireAfterWrite for cache expiration in all cache usages
- EIDAS-JCache-Dev-Node
 - added new beans timestampCacheConnectorImpl and timestampCacheProxyImpl with properties expireAfterWrite=1860 and maximumSize=10000 to the connectorCacheProvider and proxyCacheProvider beans in jCacheImplNodeBeans.xml
- EIDAS-JCache-HzIcast
 - updated iterator logic to return a functional Cache.Entry iterator over the internal ConcurrentMap to support flow scanning with Hazelcast in JCacheConcurrentMapAdapter.java

- updated Hazelcast version to 5.6.0 in pom.xml
- EIDAS-JCache-HzIcast-Node
 - updated connectorCacheProvider and proxyCacheProvider with exchangeFlowConnector and exchangeFlowProxy entries in jCacheImplNodeBeans.xml
 - added new AuthenticationExchangeFlowSerializer to enable compact serialization of AuthenticationExchangeFlow objects for hazelcast distributed caching
 - added eidas-logging module as a dependency in pom.xml
- EIDAS-JCache-Ignite
 - added AuthenticationExchangeFlow class to the ignite-whitelist.txt
- EIDAS-JCache-Ignite-Node
 - updated connectorCacheProvider and proxyCacheProvider with exchangeFlowConnector and exchangeFlowProxy entries in jCacheImplNodeBeans.xml
- EIDAS-Logging
 - updated logMessage method in several classes to replace the call of trackMessageFlow with the new markLogMessage method
 - added crossBorderLocation field to MessageLog.java
 - added new class AuthenticationExchangeFlowService containing markLogMessage method to manage authentication flow tracking, timestamps, durations, and status updates
 - added MessageOperation enum that defines the log points for connector and proxy
 - renamed MessageLoggerUtils.java class to MessageLoggerService and updated all its references
 - added MessageVector class and refactored logger methods to use it, updated setMessageVector signature and introduced applyMessageVectorToBuilder to map fields to the log builder
 - added new interface [AuthFlowPersistenceService](#) with implementations [AuthenticationFlowInfluxWriter](#), [CSVAuthFlowPersistenceService](#) and [DelegatingAuthFlowPersistenceService](#) class responsible for persisting AuthenticationExchangeFlow entries into InfluxDB and CSV
 - added new AuthenticationExchangeFlowCleanupWorker class responsible for scanning cached authentication flows and finalizing them as request error, response error, or lost based on correlation-cache expiry
 - added influxdb-client-java dependency to pom.xml
- EIDAS-Node-Connector
 - added new bean authenticationExchangeFlowCache created through getCache method in applicationContext.xml
 - added new bean authenticationExchangeFlowService with properties messageLoggerService and authenticationExchangeFlowCache, and updated messageLoggerService bean to delegate flow tracking to it in applicationContext.xml
 - updated setMessageVector method in several logging classes to set the CrossBorderLocation value
 - added influxDBClient and authenticationFlowInfluxWriter beans for writing flow data to InfluxDB to applicationContext.xml

- added new statusBoardService bean for querying authentication flow data from InfluxDB using the configured connector bucket in applicationContext.xml
- added new authenticationExchangeFlowCleanupWorker bean and enabled scheduled execution via <task:annotation-driven/> in applicationContext.xml
- added statistics.influx.enabled and statistics.csv.enabled properties to define default enablement flags for usage statistics and CSV based authentication flow logging in eidas.xml
- added example InfluxDB connection properties (influx.url, influx.token, influx.org, influx.bucket) to the default eidas.xml
- added the eidas-status-board module as a dependency in pom.xml
- EIDAS-Node-Proxy
 - added new bean authenticationExchangeFlowCache created through getCache method in applicationContext.xml
 - added new bean authenticationExchangeFlowService with properties messageLoggerService and authenticationExchangeFlowCache, and updated messageLoggerService bean to delegate flow tracking to it in applicationContext.xml
 - updated setMessageVector method in several logging classes to set the CrossBorderLocation value
 - added influxDBClient and authenticationFlowInfluxWriter beans for writing flow data to InfluxDB to applicationContext.xml
 - added new statusBoardService bean for querying authentication flow data from InfluxDB using the configured proxy bucket in applicationContext.xml
 - added new authenticationExchangeFlowCleanupWorker bean and enabled scheduled execution via <task:annotation-driven/> in applicationContext.xml
 - added statistics.influx.enabled and statistics.csv.enabled properties to define default enablement flags for usage statistics and CSV based authentication flow logging in eidas.xml
 - added example InfluxDB connection properties (influx.url, influx.token, influx.org, influx.bucket) to the default eidas.xml
 - added the eidas-status-board module as a dependency in pom.xml
- EIDAS-Parent
 - added influxdb-client-java dependency version 7.4.0 to pom.xml
 - added the eidas-status-board module as a demo tool dependency, introduced the commons-csv dependency, and included the Status Board module in the demo tools build configuration in the parent pom.xml
- LOG_HOME
 - added the logs eIDASNodeConnectorAuthFlow.csv and eIDASNodeProxyAuthFlow.csv, flat files that dump the usage statistics as CSV to be picked up by data tools or analysts.

4.3.3 Configuration changes

- EIDAS-Config
 - server/connector
 - added timestampCacheConnector cache configuration to hazelcastNode.xml

- added timestampCacheConnector cache configuration to igniteNode.xml
- added new InfluxDB connection properties using the keys influx.url, influx.token, influx.org, and influx.bucket in eidas.xml
- server/proxy
 - added timestampCacheProxy cache configurations to hazelcastNode.xml
 - added timestampCacheProxy cache configurations to igniteNode.xml
 - added new InfluxDB connection properties using the keys influx.url, influx.token, influx.org, and influx.bucket in eidas.xml

4.4 More contacts

The eIDAS Node now supports configuring multiple contacts in the metadata, in accordance with the SAML and eIDAS specifications. Previously, the configuration allowed only one support contact and one technical contact, defined through properties in eidas.xml. With this improvement, operators can configure multiple contacts for each contact type. Contacts are now defined in a dedicated xml configuration file (contacts.xml), validated against the contact-settings-1.0.xsd schema. This provides better structure, validation, and editor assistance when defining contacts. The following contact types are supported: support, technical, administrative, billing and other. At least one support and one technical contact must be configured, as required by the eIDAS specifications. Configured contacts are published in the metadata as <md:ContactPerson> elements. The contacts configuration is validated against its XSD during application startup. If the XML file exists but contains invalid configuration values, the node will fail to start and display a validation error in the logs. If the XML file is not present, the node falls back to the legacy configuration properties for support and technical contacts.

4.4.1 Code changes

- EIDAS-Metadata
 - added TypedContactData.java to represent a contact together with its contact type
 - added a new contacts variable to store the list of configured contacts in EidasMetadataParameters.java
 - added a new contacts variable to store the list of configured contacts during metadata generation in EidasMetadata.java
 - added getContacts and setContacts to expose the configured contact list through EidasMetadataParametersI.java
 - added XmlConfigLoadException to represent errors occurring during xml configuration loading and validation
 - added JaxbXmlConfigLoader to provide a generic JAXB based loader for xml configuration files with xsd validation
 - added ContactsConfigLoader.java to load contacts.xml, validate it against the XSD, and convert the configured entries into typed contacts
 - added contact-settings-1.0.xsd under src/main/resources/xsds to define and validate the contacts XML structure
 - added the xjc maven plugin configuration to generate JAXB classes from contact-settings-1.0.xsd in pom.xml
 - updated ContactData added syntactic sugar concerning expected null values in line with xsd

- EIDAS-Node-Connector
 - updated the metadata generation flow to load the full list of contacts and pass them to the metadata parameters, and updated the contact assignment logic accordingly in EidasNodeMetadataGenerator.java
 - added createConnectorContacts method to build connector contacts from XML configuration with fallback to legacy properties in NodeMetadataUtil.java
 - updated the connectorMetadataGeneratorSP bean to inject the connectorContactsConfigLoader used to load the contacts configuration in applicationContext.xml
 - added connectorContactsConfigLoader bean to initialize and validate the contacts configuration during startup in applicationContext.xml
 - updated ConnectorErrorServlet to display either eidas.xml or first support type contact in case error page is displayed
- EIDAS-Node-Proxy
 - updated the metadata generation flow to load the full list of contacts and pass them to the metadata parameters, and updated the contact assignment logic accordingly in EidasNodeMetadataGenerator.java
 - added createServiceContacts method to build proxy contacts from XML configuration with fallback to legacy properties in NodeMetadataUtil.java
 - updated the serviceMetadataGeneratorIDP bean to inject the serviceContactsConfigLoader used to load the contacts configuration in applicationContext.xml
 - added serviceContactsConfigLoader bean to initialize and validate the contacts configuration during startup in applicationContext.xml
 - updated ProxyServiceErrorServlet to display either eidas.xml or first support type contact in case error page is displayed

4.4.2 Configuration changes

- EIDAS-Config
 - server/connector/contacts
 - added contact-settings-1.0.xsd that validates the contacts configuration against SAML/eIDAS rules and provides editor support through schema-based validation
 - added contacts.xml that allows operators to configure multiple contacts per type in a structured and readable format
 - removed the deprecated legacy contacts example from eidas.xml
 - server/proxy/contacts
 - added contact-settings-1.0.xsd that validates the contacts configuration against SAML/eIDAS rules and provides editor support through schema-based validation
 - added contacts.xml that allows operators to configure multiple contacts per type in a structured and readable format
 - removed the deprecated legacy contacts example from eidas.xml

4.5 Update expired and almost expired certificates

For the example configuration, regenerated certificates and keys in p12 keystores and saml engine config files.

4.5.1 Code changes

EIDAS-Encryption

- Added a keystore skeletonKeystore.p12 to store the private keys that were used to generate the example configurations' certificates, this is to ease making changes without adding private keys to keystores that should not have private keys as part of the example.

4.5.2 Configuration changes

EIDAS-Config/server/connector/keystore

- eidasKeyStore.p12
- eidasKeyStore_METADATA.p12
- eidasKeyStore_METADATA_EC.p12
- eidasKeyStore_METADATA_TC.p12
- eidasKeyStore_METADATA_TC_EC.p12
- eidasTrustStore.p12
- EncryptModule_Connector.xml
- SignModule_Connector.xml
- SignModule_Connector_EC.xml
- SignModule_Connector_HSM_P12.xml

EIDAS-Config/server/proxy/keystore

- eidasKeyStore.p12
- eidasKeyStore_METADATA.p12
- eidasKeyStore_METADATA_EC.p12
- eidasKeyStore_METADATA_TC.p12
- eidasKeyStore_METADATA_TC_EC.p12
- eidasTrustStore.p12
- SignModule_Service.xml
- SignModule_Service_EC.xml
- SignModule_Service_HSM_P12.xml

4.6 Upgrade tomcat version to latest 11 patch version

The tomcat-embed-core dependency used by the application was updated to 11.0.20 for security updates. The recommended Tomcat application server version was also updated to 11.0.20 version.

4.6.1 Code changes

- EIDAS-Parent
 - updated tomcat-embed-core dependency to 11.0.20

4.7 Other changes requiring no action

4.7.1 Upgrade apache commons-lang to commons-lang3

The Apache Commons Lang library was upgraded from version 2.6 to 3.19.0.

4.7.1.1 Code changes

- EIDAS-Parent
 - upgraded commons-lang 2.6 to commons-lang3 3.19.0 in pom.xml
 - updated several classes to use org.apache.commons.lang3 imports instead of org.apache.commons.lang in modules EIDAS-Commons, EIDAS-Encryption, EIDAS-Light-Commons, EIDAS-Logging, EIDAS-Node-Connector, EIDAS-Node-Proxy, EIDAS-SAMLEngine, EIDAS-Metadata, EIDAS-Security and EIDAS-SpecificCommunicationDefinition

4.7.2 Upgrade logback-classic dependency version

Upgraded ch.qos.logback:logback-classic:1.5.18 to 1.5.32 version.

4.7.2.1 Code changes

Updated EIDAS-Parent/pom.xml with `<logback.version>1.5.32</logback.version>`

4.7.3 Upgrade spring to 6.2.13

Upgraded org.springframework:spring-core and other spring modules from 6.2.9 to 6.2.13 because of CVE-2025-41249

Upgraded io.micrometer:micrometer-core from 1.14.5 to 1.14.13 because of spring 6.2.13 patch notes.

4.7.3.1 Code changes

Updated EIDAS-Parent/pom.xml with `<spring.version>6.2.13</spring.version>` and `<micrometer.core.version>1.14.13</micrometer.core.version>`

4.7.4 Unable to override IGNITE_MARSHALLER_WHITELIST due to improper default mechanism (EIDSD-1020, EIDSD-1007)

When adding a default whitelist securing the data class whitelist for ignite 2.17 in 3.0.0 we introduced a bug that made it impossible to override the IGNITE_MARSHALLER_WHITELIST with anything useful.

This impacts installations where other application make sure of Ignite and are deployed in the same environment.

4.7.4.1 Code changes

Updated IgniteUtils to only consider a missing Environment or Java Parameter as a reason to set the default value.

Added IgniteUtilsTest with use cases to verify the intended behavior

Added JUnit5 and system-stubs-jupiter to Ignite implementation

4.7.5 Set CacheManagers to Lazy

The cache implementation beans `connectorCacheProvider`, `proxyCacheProvider`, `connectorSpecificCacheProvider`, `proxySpecificCacheProvider` will no longer be instantiated if there is no use of them in the application that uses them, this prevents half of them to be instantiated in connector or proxy. This for the included implementations of Ignite, Hazelcast, in-memory cache.

4.7.5.1 Code changes

- `eidas-jcache-dev-node`
 - `jCacheImplNodeBeans.xml`
- `eidas-jcache-dev-specific-communication`
 - `jCacheImplSpecificCommunicationBeans.xml`
- `eidas-jcache-hazelcast-node`
 - `jCacheImplNodeBeans.xml`
- `eidas-jcache-hazelcast-specific-communication`
 - `jCacheImplSpecificCommunicationBeans.xml`
- `eidas-jcache-ignite-node`
 - `jCacheImplNodeBeans.xml`
- `eidas-jcache-ignite-specific-communication`
 - `jCacheImplSpecificCommunicationBeans.xml`
- `eidas-jcache-dev`
 - `jCacheImplSpecificCommunicationBeans.xml`