CEF eID eIDAS-Node Migration Guide

Version 1.3

Document history

Version	Date	Modification reason	Modified by
1.3	09/08/2017	Origination	DIGIT

Table of contents

List of tables

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 the document is to facilitate migration from an older implementation of the eIDAS-Node to eIDAS-Node v1.3.

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 1.3.
- Chapter 3 — *Changes*: Contains detailed information about the changes that should be taken into consideration when migrating to eIDAS-Node version 1.3.

1.2. Document aims

The main aim of this document is to provide information on all the changes requiring your action when migrating to eIDAS-Node version 1.3, including:

- configuration changes; and
- changes to common code.

2. Prerequisites

Before starting your migration to eIDAS-Node version 1.3 you should have:

- already implemented eIDAS-Node version 1.2 or older;
- downloaded the eIDAS-Node v1.3 Integration Package; and
- downloaded the latest documentation.

3. Changes

3.1. Summary of changes

In eIDAS-Node version 1.3 there are several changes that affect your installation. The main changes are:

- Configuration for skew time: changes to the way optional skew time is interpreted (see section 3.2);
- Configuration is now externalised, most of the configuration files can be moved to an external directory on the file system (see section 3.3);
- Adjustments have been made to the `LightRequest` interface (see section 3.4);
- Corrections to logging (see section 3.5);
- Support for sha256-rsa-MGF1 (signature) (see section 3.6);
- The 'updater' service (simple context path driven reloading of application context) has been disabled by default (see section 3.7);
- Validation of optional attributes in eIDAS-Node Proxy Service (see section 3.8);
- Changes to backward compatibility of `SubjectLocality`, `SubjectConfirmationData` and `OneTimeUse` elements (see section 3.9); and
- Improvements to make the Metadata generator more robust (see section 3.10).

3.2. Configuration for skew time adjustment

Table 1: Configuration for skew time adjustment — Quick reference of changes

Subject	Item	Description
Node	*eidas.xml*	Add `service[n].skew.notbefore` and `service[n].skew.notonorafter` added to *eidas.xml* Remove obsolete `service[n].skew`
	Code changes	Connector - call to validation of time conditions with new configuration parameters
ProtocolEngine	Code changes	Removed OpenSAML time conditions validation suite, adjusted existing check code

In previous versions it was not possible to apply server skew of `notBefore` and `notOnOrAfter`, because the OpenSAML validation suite has not used the `eidas.xml` provided setting. The solution is re-implemented with the new variables introduced to `eidas.xml`:

- `service[n].skew.notbefore`: set this variable to add specified milliseconds when validating `notBefore` condition of SAML response. Also accepts negative values (e.g. `service3.skew.notbefore`=-3600).

- `service[n].skew.notonorafter`: set this variable to add specified milliseconds when validating `notOnOrAfter` condition of SAML Response (e.g. `service3.skew.notonorafter`=3600). Also accepts negative values.

Remove any obsolete configuration from `eidas.xml` named `service[n].skew`.

The main goal of skew is to shift the condition validation based on an experienced clock skew value. The service skew can be used with negative `notbefore` and positive `notonorafter` to enlarge the validation frame, however it is not recommended. Clocks should be synchronised between servers responsible for authentication.

3.3. Externalised configuration

Table 2: Externalised configuration adjustment — Quick reference of changes

Subject	Item	Description
Node	Location of configuration files	Location of these files is learned from environment variable(s).
	Application context	Added new `defaultPath` parameters to `ProtocolEngineFactory`
	Code changes	Propagation of a `defaultPath` through the configuration accessors
ProtocolEngine	Location of configuration files	Location of these files is picked up from environment variable(s).
Demo Tools	Location of configuration files	Location of these files is picked up from environment variable(s).

3.3.1. Configuration changes

Loading of configuration files has been refactored in eIDAS-Node version 1.3. There are new environment variables introduced in order to specify the file locations:

- `EIDAS_CONFIG_REPOSITORY`: for files related to the eIDAS-Node (`eidas.xml`, `hazelcast.xml`) and also used in ProtocolEngine configuration too.
- `SPECIFIC_CONFIG_REPOSITORY`: for files related to sample MS-Specific part of the eIDAS-Node (`eidas_Specific.xml`) and used in ProtocolEngine configuration too.
- `SP_CONFIG_REPOSITORY`: for Demo SP configuration (`sp.properties`) and ProtocolEngine settings, including attribute registry (`saml-engine-eidas-attributes.xml` and `saml-engine-additional-attributes.xml`).
- `IDP_CONFIG_REPOSITORY`: for Demo IdP configuration (`idp.properties`, `user.properties`) and included ProtocolEngine settings.

These variables can be provided via Operating System or Application Server environment defined variables or by specifying –D option to JVM (e.g. `-D EIDAS_CONFIG_REPOSITORY=/home/opt/eidas/config`).

ProtocolEngine configuration is also read from these locations, so they can be kept together with module configuration. It is good practice to introduce different folders for each module (Node, Specific, IDP, SP). There are some internal path elements in `ProtocolEngine` configuration XMLs. All of these are now relative to the location pointed to by REPOSITORY variables (and therefore the referring XMLs). Configuration not related to ProtocolEngine (`eidas.xml`) still refers to absolute path (e.g. Metadata file repository).

It is possible to set the hard coded path in `environmentContext.xml` if necessary.

3.3.2. Implementation changes

The way the ProtocolEngine is initialised has been changed. When creating an instance (programmatically or by `environmentContext` Spring injection) there is an extra parameter needed, called `defaultPath`. If "null" supplied, the behaviour is the same as in the previous versions, the file will be loaded from the `CLASSPATH`, however the internal path variables will be relative.

The references from `environmentContext.xml` are added to `applicationContext.xml` and `specificApplicationContext.xml` files, so if your eIDAS-Node installation has customised any of these, you need to adjust the parameters of ProtocolEngineFactory (see sample bundled file).

3.4. SPType in LightRequest

Table 3: SPType in LightRequest — Quick reference of changes

Subject	Item	Description
Node	Code changes	Propagate `SPType` to Specific
Specific interface	Fields	Added new "`SPType`" field to `LightRequest`
Demo Tools	Configuration	Added option `sp.type` to `sp.properties` to display `SPType` in Metadata of SP. Undefined if omitted.

Adjustments have been made to the `LightRequest` interface to enable the `SPType` element to be propagated from MS-Specific Connector to eIDAS-Node Connector, and then from eIDAS-Node Proxy Service to MS-Specific Proxy Service.

This implementation comes with new validation rules.

- Connector validates `SPType` provided by `LightRequest` against the optional Node sector setting (`metadata.sector`) from `eidas.xml`.
- Connector adds this information to eIDAS Request only if not specified to the Node itself (and published in the Metadata).

- Proxy still validates the `SPType` in the Request against the Metadata of the Connector.

However there is one scenario when the `SPType` is not propagated to MS-Specific Proxy Service. If it is available from the Metadata of the eIDAS-Node Connector, and therefore not sent in the eIDAS Request, it will not show up in the `LightRequest`. It will be implemented in the next release.

3.4.1. Configuration changes

DEMO SP has a new setting in `sp.properties` file: `sp.type` handles the presence of `SPType` in Metadata. It can be public, private or omitted.

3.4.2. Implementation changes

`ILightRequest` implements a new String type field can be accessed with `getSpType`, and set by the any of the Request builders.

3.5. Corrections to logging

Table 4: Corrections to logging — Quick reference of changes

Subject	Item	Description
Logging		`OpType` field is corrected
Node	Code changes	Dependency on Specific logger was removed.

There are two fixes introduced:

- The Audit Log contains a field named `OpType` that was not properly logged. There was a timestamp here, instead of the real `opType` information.
- There was a dependency in the eIDAS-Node, as it was using `eu.eidas.node.auth.specific.EidasLoggerBean` from sample MS-Specific package. The logger code was moved to the eIDAS-Node (`eu.eidas.node.logging.EidasLoggerBean`) the application context files were cleared up.

3.6. Support for sha256-rsa-MGF1 (signature)

Table 5: sha256-rsa-MGF1 (signature) — Quick reference of changes

Subject	Item	Description
Node	Code changes	Extended the list of supported algorithms, added new digest mapping
Node	Configuration	Extended whitelist for Metadata

Subject	Item	Description
Protocol Engine	Configuration	Extended whitelist for Protocol Engine

Added support for sha256-rsa-MGF1 as the Cryptographics Requirements part of eIDAS Technical Specification permits its usage.

3.6.1. Configuration changes

All the `SignModule_<enginename>.xml` and `eidas.xml/eidas_Specific.xml` files were updated with the extended whitelist, please check the samples.

3.6.2. Implementation changes

The hardcoded whitelist of the eIDAS-Node was extended with this algorithm, with mapping to SHA256 digest. Checks for the identifiers are now case-insensitive.

3.7. Updater is disabled

Table 6: Updater is disabled — Quick reference of changes

Subject	Item	Description
Node	Configuration	Disabled Updater Servlet in web.xml

The 'updater' service (simple context path driven reloading of application context) has been disabled by default in `web.xml`, because of not-appropriate check on invocation. It will be replaced by sophisticated administration possibilities in later releases, but the service can still be re-enabled.

3.8. Validation of optional attributes in Proxy Service

Table 7: Validation of optional attributes in Proxy Service — Quick reference of changes

Subject	Item	Description
Node	Code changes	Corrected check of "required" flags of attributes in incoming Request

eIDAS-Node version 1.1 introduced a regression with check of 'required' flags in eIDAS-Node Proxy Service. When a Request arrived, check had not been made on this flag against the `AttributeRegistry`. Version 1.3 re-implements this check with all the attributes, including the additional ones.

3.9. SubjectLocality, SubjectConfirmationData and OneTimeUse elements

Table 8: SubjectLocality, SubjectConfirmationData and OneTimeUse elements — Quick reference of changes

Subject	Item	Description
Node	Code changes	Extended the list of supported algorithms, added new digest mapping

The following changes eliminate backward compatibility on optional `SubjectLocality` and `OneTimeUse` elements introduced in eIDAS-Node version 1.2.

- `OneTimeUse` is no longer generated to the Response (it can be re-enabled in `DefaultCoreProperties.java`, setting `oneTimeUse` to 'true').
- `SubjectLocality` is no longer generated in the Response.

In addition:

- IP address (of web client) in `SubjectConfirmationData` is generated only if there is an IP address supplied in the `LightResponse`. However that IP address is discarded and the eIDAS-Node Proxy Service replaces it with the IP address of the web client accessing the eIDAS-Node.

The removal of the optional SAML elements will end compatibility between eIDAS-Node version 1.3 and version 1.1 (and also any other not-supported previous versions).

3.10. Improvements in Metadata generator/caching

Table 9: Improvements in Metadata generator/caching — Quick reference of changes

Subject	Item	Description
Node	Code changes	Refactorings in Metadata generator
Node	Configuration	New, mandatory organisation data variables in eidas.xml, retention period for internal cache
Demo Tools	Configuration	Added retention period for Metadata cache

There are some refactoring improvements in version 1.3 to make the Metadata generator more robust. Since the `OrganizationName` element was not displayed, during the refactoring all the Organization Metadata were collected together in `eidas.xml`.
The retention time of internal, default cache is now configurable.

3.10.1. Configuration changes

The new variables are mandatory:

- `connector.organization.name`, `connector.organization.displayname`, `connector.organization.url`,
- `service.organization.name`, `service.organization.displayname`, `service.organization.url`.

If the internal, default Metadata cache is used, the new `nonDistributedMetadata.retention` must be set in `eidas.xml`.

Please check provided sample configuration for more information.

If testing with Demo SP or IdP, the configuration in `sp.properties` and `idp.properties` must be extended with: `organization.name`, `organization.displayname`, `organization.url` elements.

There is an improvement to Metadata consumption of Demo SP and Demo IdP: a simple caching service was introduced. Therefore the new `sp.metadata.retention`/`idp.metadata.retention` variables must be set to the according properties file.