Required certificates, key- and truststores in the e-Codex environment

Table of contents

TABLE OF CONTENTS 2

1. INTRODUCTION 3

1.1. SCOPE AND OBJECTIVE OF THIS DOCUMENT 3

1.2. TRANSPORT LAYER SECURITY (TLS/SSL) 3

1.3. THE GATEWAY COMPONENT 3

1.4. TYPES OF CERTIFICATES/KEY PAIRS 3

1.5. HOW TO GATHER CERTIFICATES/KEY PAIRS 3

2. OVERVIEW 5

2.1. ARCHITECTURAL OVERVIEW 5

2.2. MULTI-BACKEND EXAMPLE 5

2.3. REQUIRED KEY PAIRS 6

3. BACKEND SECURITY 7

3.1. SECURITY LEVEL AT THE BACKEND OF THE DOMIBUSCONNECTOR 7

3.2. THE CONNECTOR CLIENT KEYSTORE 8

4. SECURITY ON THE DOMIBUSCONNECTOR LEVEL 9

4.1. CONNECTOR BACKEND KEYSTORE 9

4.2. CONNECTOR EVIDENCES KEYSTORE 10

4.3. CONNECTOR SECURITY KEYSTORE 10

4.4. CONNECTOR SECURITY TRUSTSTORE 10

5. GATEWAY SECURITY 12

6. E-CODEX CONFIGURATION MANAGEMENT 13

1. Introduction

1.1. Scope and Objective of this document

As there are several levels of security within the e-Codex environment, there is also the requirement to use different certificates to fulfil those security requirements. This document explains in examples what certificates, keys and stores are used within the e-Codex building blocks. It also explains some basic security features and what the purposes of those are.

After reading this document clarity should be given on what certificates are required, how to gain them and which stores should hold them.

1.2. Transport layer security (TLS/SSL)

This document does not describe in detail how the transport layer security (TLS/SSL) works. This is mostly dependent on the infrastructure used and mostly terminated by the web containers or web servers. With applications not running in web servers, like the "domibusConnectorClient-Standalone", the Java Runtime Environment (JRE) handles the certificate handling for transport layer security. Therefore a basic knowledge on TLS/SSL security mechanisms should be given.

1.3. The gateway component

In this document also keys and stores used within the gateway are described. As an example for the gateway, the e-Codex gateway "domibus" is used.

There are other AS4/ebms3 compliant gateway vendors available that can be used in the e-Codex environment.

While describing the usage and purposes of stores and keys on the gateway level, this document does not describe how to install and configure them on the gateway.

Please use the documentation of the gateway vendor used within your environment.

For the domibus gateway such documentation can be found following this link:

https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Domibus

1.4. Types of certificates/key pairs

Software security is a fast living topic.

Therefore also requirements for certificates are changing very fast. To be a long term useable documentation, this document does not describe types of certificates or key pairs.

1.5. How to gather certificates/key pairs

Within this document mostly self-signed certificates and key pairs are used as examples.

Whereas this is sufficient for the purpose of documentation, this is mostly not sufficient when it comes to exchange messages over the internet, or even over the intranet (depending on your infrastructure and security policies).

This document shows how to use self-signed certificates and key pairs. It does NOT describe how you can gather certificates from trusted certification authorities.

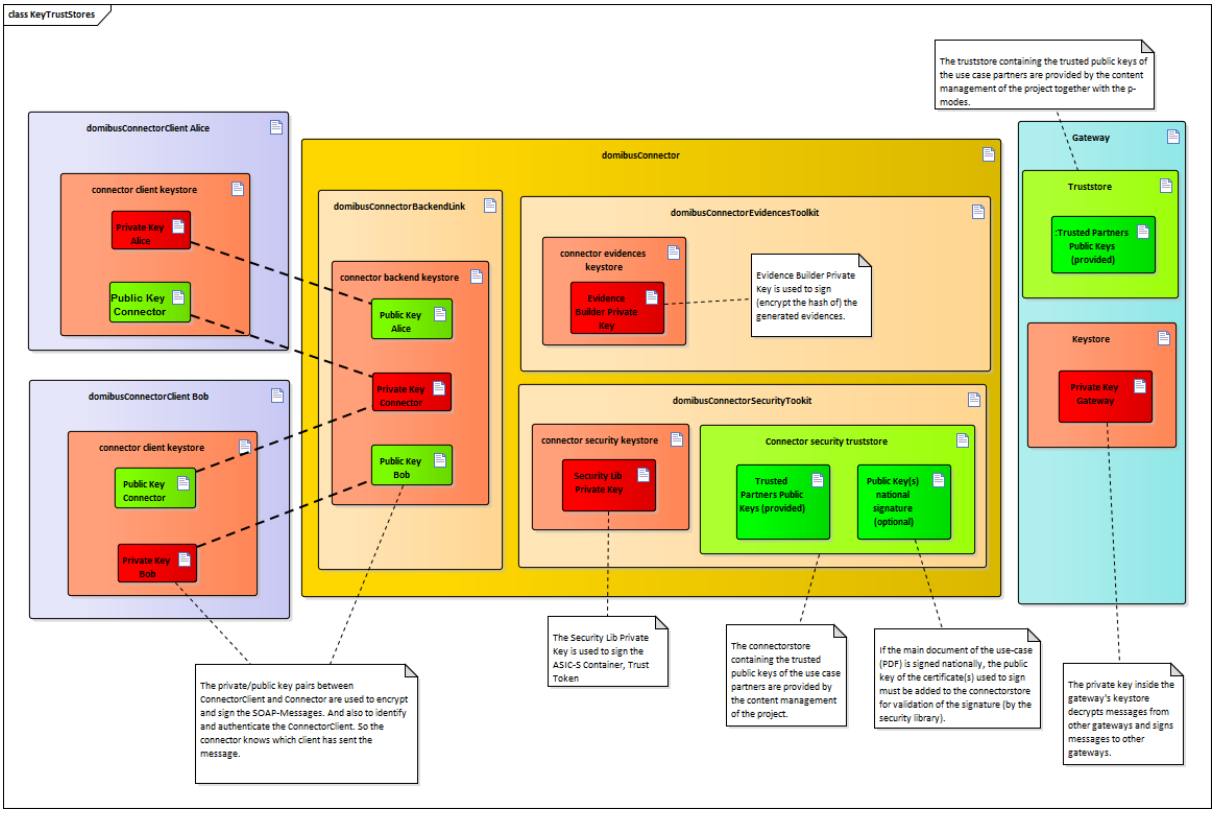
2. Overview

This chapter gives an overview on what is described in detail within this document.

Also the examples used in this document are introduced.

2.1. Architectural overview

Further descriptions within this document will rely on the following architecture:



- Gateway: As already explained in chapter The gateway component, this document will describe used stores and keys on the gateway level.
- domibusConnector: The domibusConnector as a web application will need the most stores and keys for handling the message security.
- domibusConnectorClient: As examples for backend applications, the domibusConnectorClient will be used.

2.2. Multi-backend example

To show how the security between the domibusConnector and backend clients works, the introduced example in this document has two backend applications in place. They are called “Alice” and “Bob”.

Both are domibusConnectorClient implementations in this example. The names should help to seperate the two backends from each other.

2.3. Required key pairs

Key pairs are a couple of a private and a public key. While the public key must be shared with the respective communication partner (which in this documentation could also be another component) the private key (from the view of the component or partner using the key pair) should never be shared. Going through the process from the back to the front, the following key pairs are required:

- **Backend key**: Every backend requires a separate key pair. The keys are used to encrypt/decrypt messages when exchanged with the domibusConnector. They are also used for signing the messages in exchange. Furthermore the backend certificate is also used to configure the backend on the domibusConnector and authenticate the backend when it connects with the domibusConnector.
- **Connector backend key**: This key pair is the counterpart of the backend key pair. It also signs and encrypts/decrypts messages exchanged with each backend.
- **Connector evidences key**: While processing messages, the domibusConnector at a certain point creates and distributes confirmations on the status of the message processing. The domibusConnector uses ETSI-REM evidences for this purpose. When creating such evidences a hash code of the main document of the message is created. To encrypt this hash code also a private key is used.
- **Connector security key**: The domibusConnector has a security toolkit integrated that packs and signs the documents and attachments of messages together and signs the created container. This container is called the “ASIC-S” container. To create the signature of that container the connector security certificate’s private key is used.
- **Gateway key**: Exchanging messages between AS4 compliant gateways require, dependent on the configuration agreed upon per use-case, that messages are signed and/or encrypted on different levels. For that purpose the gateway also needs a key pair.

We strongly recommend using different key pairs for all those different purposes.

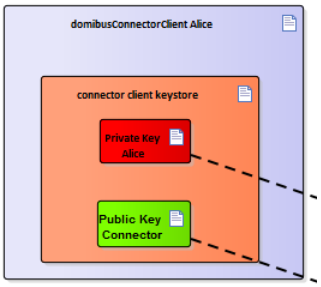
Anyway, besides the backend key(s) which must definitely be different ones, all other keys may be the same one.

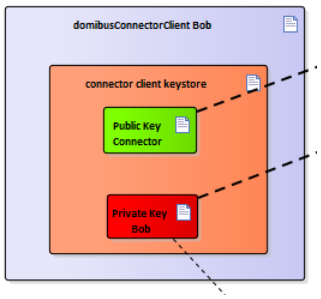
3. Backend security

From version 4.0.0 onwards, the domibusConnector is capable to handle multiple backend applications.

To be able to authenticate the backend connecting to the domibusConnector, and also to handle message security between the backend and the domibusConnector, every backend needs a certificate and a keystore.

The example used in this document and introduced in the chapter Overview uses two backend applications connecting to the domibusConnector: Alice and Bob.





Let's assume that both of them own a certificate. The certificate for backend Alice also has "Alice" as its common name (CN) defined. This is important for configuring backend Alice in the domibusConnector database. Also the certificate for backend Bob has the CN "Bob" defined.

Also, to handle security with the domibusConnector, the public key of the connector backend certificate is required. This is one certificate used with every backend by the domibusConnector.

3.1. Security level at the backend of the domibusConnector

The web service interfaces used between the backend(s) and the domibusConnector are defined as SOAP messages using the "ws-security-policy" standard by OASIS.

This implies that messages exchanged over those interfaces are SOAP messages that are signed by the sender of the message. This signature is validated by the receiver of the message.

It further means that the messages' header and contents are encrypted.

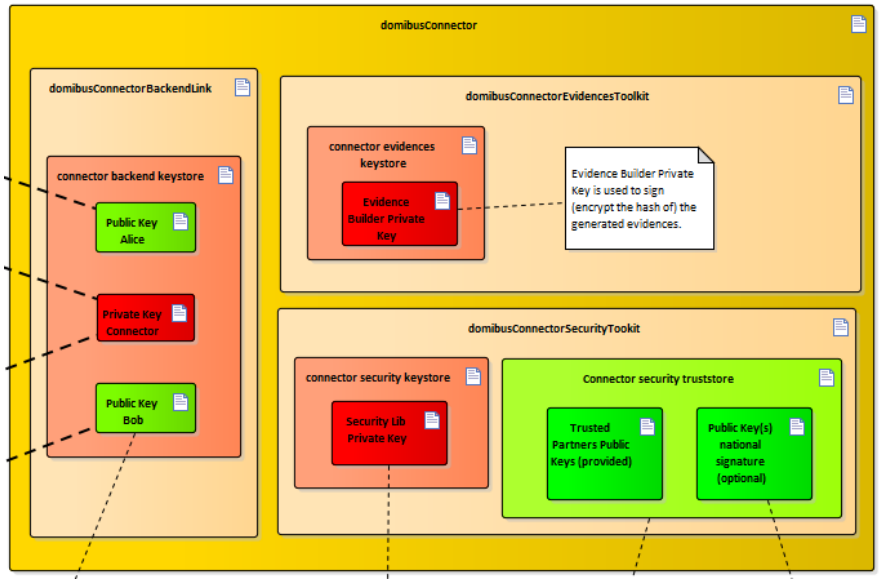
3.2. The connector client keystore

Two Java-Keystores have to be created:

- Connector client keystore “Alice” contains:
  - The public key of the domibusConnector backend certificate. It is used to validate signatures of messages received from the domibusConnector and also to encrypt messages that are sent to the domibusConnector.
  - The private key of the certificate “Alice”. It is used to sign messages that are sent to the domibusConnector and also to decrypt messages received from the domibusConnector.
- Connector client keystore “Bob” contains:
  - The public key of the domibusConnector backend certificate. It is used to validate signatures of messages received from the domibusConnector and also to encrypt messages that are sent to the domibusConnector.
  - The private key of the certificate “Bob”. It is used to sign messages that are sent to the domibusConnector and also to decrypt messages received from the domibusConnector.

4. Security on the domibusConnector level

The domibusConnector itself requires the most key- and truststores:



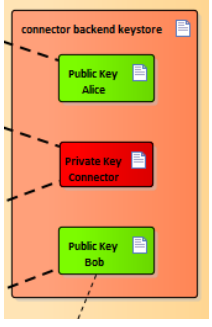
The reason for this is that the domibusConnector covers the workflow process of a business message. This means handling of the documents, attachments, routing information and also creating and handling of the confirmation messages, in the case of the domibusConnector ETSI-REM evidences.

Additionally the domibusConnector administers information on the backend(s) connecting to the domibusConnector.

4.1. Connector backend keystore

Just like at the backend side, described in chapter Backend security, the domibusConnector also needs a keystore holding certificates to handle security with the backend(s).

Also here this document refers to the introduced example of two backend applications connecting to the domibusConnector: “Alice” and “Bob”.



Therefore the connector backend keystore must contain the following keys:

- The private key of the domibusConnector backend certificate. It is used to sign messages that are sent to the backends “Alice” and “Bob” and also to decrypt messages received from the backends.
- The public key of backend “Alice”. It is used to validate signatures of messages received from “Alice” and also to encrypt messages that are sent to the backend “Alice”.
- The public key of backend “Bob”. It is used to validate signatures of messages received from “Bob” and also to encrypt messages that are sent to the backend “Bob”.

4.2. Connector evidences keystore

As already described, the domibusConnector generates confirmations at different points in message processing to distribute information to the sender/receiver of the business message on the status of the message.

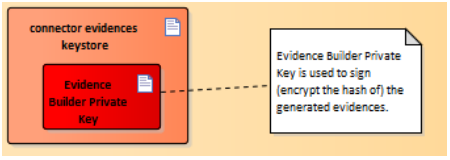
In the case of e-Codex the ETSI-REM standard is used to produce evidences.

Those evidences contain information on the document of the business message that can be validated at any time to proof that the document is still the same as when creating the message.

For this purpose a hash code of the main document is generated and put into the ETSI-REM evidence. The evidence furthermore is signed by the domibusConnector.

This signature has to be created using a certificate. In chapter Required key pairs it is called “connector evidences certificate”.

The private key of this certificate has to be placed inside a Java-Keystore:

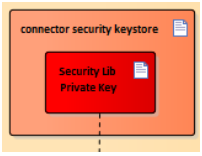


4.3. Connector security keystore

It is part of the workflow for business messages that the main document of the message together with most of the attached documents are packed to a secure container.

This secure container gets signed and is then a so called “ASIC-S” container.

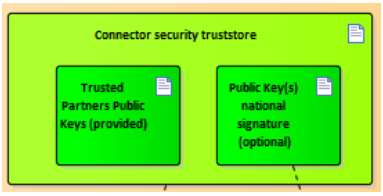
To create the signature of the ASIC-S container, a certificate is needed. In chapter Required key pairs it is called “connector security certificate”.



The private key of this certificate needs to be added to the “connector security keystore”.

4.4. Connector security truststore

The connector security truststore is mostly provided by the e-Codex configuration management.



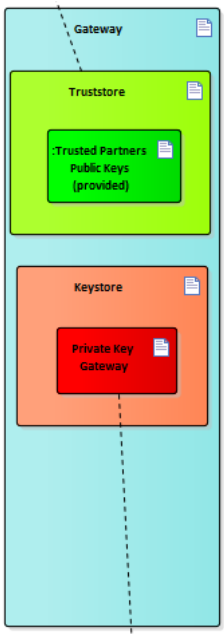
This truststore only holds public keys. The connector truststore (in configuration management called the “connectorstore”) is provided by the configuration management of the project and contains the public keys of the e-CODEX partners. They are used to verify the signature of the ASIC-S container, previously described, received from an e-CODEX partner.

Additionally, if your organization uses signed documents (mostly PDF) as the main document of the business message when sending a message to an e-CODEX partner, the public key of the certificate with which the document was signed with should be imported into this truststore. The security library uses this public key to verify the signature of the document then (configured as SIGNATURE_BASED).

5. Gateway security

Within e-Codex it is defined that gateways to exchange messages must be AS4 compliant. The AS4 pattern is used to transport SOAP message in a secure way.

Depending on the configuration, which the e-Codex partners of a use case must agree upon, the messages exchanged are signed and encrypted.



For that purpose, a keystore has to be created, holding the private key of the certificate described as “Gateway certificate” in chapter Required key pairs.

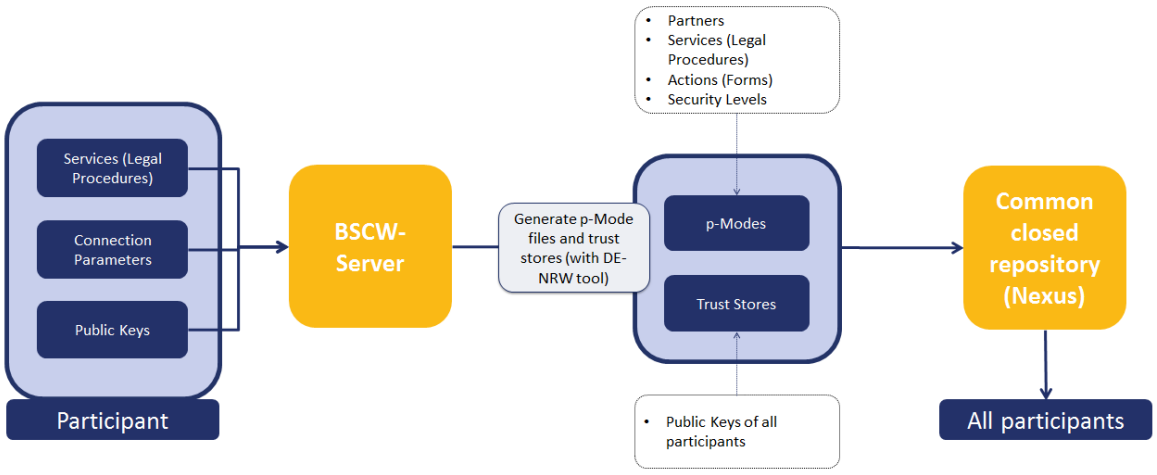
It is used to sign and decrypt messages to be exchanged with other gateways.

The truststore for the gateway, just like the “connectorstore” on domibusConnector level, is provided by the e-Codex configuration management.

6. e-Codex configuration management

Within pilots using the e-Codex environments, a central configuration management is in place.

Its purpose is to collect data on e-Codex partners in the context of a business use-case to provide this data to the other partners.



This document only focuses on the certificate related parts of the configuration management.

Every participant has to provide the public keys of the following certificates described in chapter Required certificates:

- Connector security certificate
- Gateway certificate

They are used to create truststores that contain the public keys of all partners.

The “Connector security certificate” public key is used for the “connectorstore” distributed by the configuration management. The “connectorstore” distributed contains all public keys from all participants and is described in this document as Connector security truststore.

The “Gateway certificate” public key is used for the “truststore” distributed by the configuration management. The “truststore” distributed contains all public keys from all participants and is described in this document in chapter Gateway security.

One additional certificate’s public key is needed by the configuration management that is not described in this document. It is the SSL certificate’s public key of your environment.