# eCodex

## Table of Contents

# e-Codex Protocol specification

## Abstract

e-Codex is a collection of different protocols and standards and is tying them together. One of the main goals of e-Codex is the secure and and traceable transport of digital documents. For this purpose e-Codex is built on top of AS4.

## Specifications of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## Status of this document

This document is currently work in progress and should describe e-Codex in a detailed fashion to help other developers to implement it.

## Copyright Notice

## Introduction

Within the judicial domain it becomes more likely that cases are international. A lot of countries already have national systems to manage and trace digital received documents. But this systems are unable to communicate with each other in a cross border manner. E-Codex is offering an approach to address the technical problems which are occuring in transporting documents digitally between different systems.

The judicially domain as an urgent need to always proove the authenticity of a document and also proove when the document has entered the disposal area of a specific authority.

## Message Flow

The e-Codex message flow is business case driven. Multiple business messages related to each other should grouped by the usage of the same conversationId. The conversationId must be a generic key and must not contain any sensible information.

- sending system
    - SUBMISSION_ACCEPTANCE should be created if the message is considered valid
    - SUBMISSION_REJECTION should be created if the message is not a valid e-Codex or contains any information which makes it obvious for the sending system that this message will not be accepted by the receiving system.
    - RELAY_REMMD_FAILURE should be created if the configured timeout was reached and no evidence of the type RELAY_REMMD_ACCEPTANCE, DELIVERY_ACCEPTANCE,

    DELIVERY_REJECTION, RETRIEVAL_ACCEPTANCE, RETRIEVAL_REJECTION has been received.

- receiving system
    - RELAY_REMMD_ACCEPTANCE should be created if the message is received at the new domain and can be processed and passed to the next relaying system or target system.
    - RELAY_REMMD_REJECTION should be created if the message cannot be processed or passed to the next relaying system or target system.
    - DELIVERY should be created if the message was successfully passed to the target system. The DELIVERY_ACCEPTANCE should be generated accordingly to the national law when the message has been legally passed to the receiving authority. The usage and meaning of the DELIVERY should be defined within the business case.
    - NON_DELIVERY should be created if the message cannot be passed to the target system, authority, or even when the acceptance of the message is denied due formal reasons. The usage and meaning of the NON_DELIVERY should be defined within the business case.
    - RETRIEVAL may be created if the message has been seen by the processing system, user, human, The usage and meaning of the RETRIEVAL should be defined within the business case.
    - NON_RETRIEVAL may be created if the message has not been seen by the processing system, user human within a given time period. The usage and meaning of the NON_RETRIEVAL should be defined within the business case.

## Asic-S Container Format

As the content is packaged into the ASIC-S container. The ASIC-S container format itself is described at http://www.etsi.org/deliver/etsi_ts/103100_103199/103174/02.02.01_60/ts_103174v020201p.pdf

The following documents are part of the ASIC-S container:

- A business document of any name.
- Attachments

## Token.xml format

The token must be generated by the sending e-Codex system. It must be signed with a certificate which is valid in the e-codex environment.

## Evidences

To track the message during the way through the different systems the message processing points must generate ETSI-REM evidences. The evidences are always generated and processed in the xml-format.

TODO: describe evidences in more detail, link the ETSI document.

The following evidences must be supported by an e-Codex system. The messages are distinguished in the good case and the error or fault case. For the different cases the evidences are named differently in UPPER-CASE letters according to the following list:

- SubmissionAcceptanceRejection
    - SUBMISSION_ACCEPTANCE
    - SUBMISSION_REJECTION
- RelayToREMMDAcceptanceRejection
    - RELAY_REMMD_ACCEPTANCE
    - RELAY_REMMD_REJECTION
- RelayToREMMDFailure
    - RELAY_REMMD_FAILURE
- DeliveryNonDeliveryToRecipient
    - DELIVERY
    - NON_DELIVERY
- RetrievalNonRetrievalByRecipient
    - RETRIEVAL
    - NON_RETRIEVAL

## Evidences handling

A message with an SUBMISSION_REJECTION should not leave the sending system. If a message with an SUBMISSION_REJECTION is received by an receiving system the system must ignore it. Any following messages or evidences to this message must be ignored.

Note This means that the receiving system must track at least the message metadata and evidences itself to be able to track the assocation between the messages.

## AS4 Message Format/Structure

Within e-Codex the messages are exchanged over AS4. The e-Codex messages are transported with the e-SENS AS4 profile. https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/e-SENS+AS4+-+1.12

The exchanged messages must have additionally to the e-SENS profile attributes the following structure:

The message properties finalRecipient and originalSender must be set.

There are 2 types of messages. The evidence messages and the business message.

The business message must follow the following rules:

- The message must have a payload with the name *messageContent* this content is the business xml and must have the mimeType *text/xml*

- The message must have a payload with the name *container-signed-xades-baseline-b.asics* this content is the ASIC-S container it must have the mimeType *application/vnd.etsi.asic-s+zip* and it must be a valid ASIC-S e-Codex container.

- The message must have a payload with the name *Token.xml* this content must be a valid e-Codex Token.xml and must have the mimeType *text/xml*

- The message should have a payload with the name SUBMISSION_ACCEPTANCE this content is a evidence of the type SUBMISSION_ACCEPTANCE with the mimeType *text/xml*.

The confirmationMessage must follow the following rules:

- The message must have a payload with the name of the evidence [see evidence chapter]

- The AS4 property refToMessageId must be set to the business message ebmsId the evidence was generated for.

- The confirmation message must contain only one evidence

## Certificates

Within an e-Codex environment multiple certificates are used. For a valid e-Certificate in an e-Codex environment the certificate must comply to following rules. Any references to CRL or OCSP protocol links or future X509 protocols must be reachable within the e-Codex environment. So is the e-Codex system running over a public network then the references must also be resolvable over this network.

## Example Deployment

The following setup describes an example setup of how the e-codex components can be installed.

Figure 1. Deployment Graphics

## Components

Table 1. Table Components

| Name | Description |
|---|---|
| Internet | Symbolizes the Internet, The connection between the Gateways is made through it |
| Firewall, DDOS-Filter | A Firewall which protects the internal network. The information distributed over the CMT can be used to reduce the range of IP-Addresses which can connect through this firewall to the Apache Web Server |
| Apache Web Server | The Apache Web Server in this Setup is responsible for terminating the TLS-Connection, Authenticating the connecting Web-Clients (the remote Gateway) by checking the client certificate |
| Tomcat Web Application Server - Domibus Gateway | The domibus Gateway is deployed on a Tomcat Web Application Server |
| Tomcat Web Application Server - Domibus Connector | The domibusConnector is deployed on a Tomcat Web Application Server |
| Tomcat Web Application Server - Connector Client Application | The Connector Client Application is deployed on a Tomcat Web Application Server. This can be a national solution of an eCodeX-UseCase or the ReferenceImplementation. |
| Database | The database (MYSQL) provides for each application a different database. |

```
**Note that each application is running on it's own tomcat application server in this exa
```

### Firewall, DDOS-Filter

The firewall acts in this setup as simple IP-Filtering mechanism and ensures that only the partners of the ecodex-Network can access the domibus gateway.

## Example Apache Setup and alternatives

Possible alternatives for the apache in this use case can be:

- NGINX
- IIS
- Apache
- Tomcat (using tomcat for TLS-terminiation)
- …

The most commonly used webserver is the apache webserver. http://httpd.apache.org/ Apache is well doucmented and there are a lot of examples. Important for the setup is that the tls (at least TLS 1.2 better TLS 1.3) encryption is enabled on apache. For this purpose the apache server needs a certificate. You can obtain this certificate also over letsencrypt.

As apache is the most used webserver over the whole internet the next paragraphs will cover the e-Codex specifics of an apache setup.

### Setting up Apache

The apache webserver project site offers a good tutorial for setting up apache with TLS-encryption. https://httpd.apache.org/docs/2.4/en/ssl/ssl_howto.html

After the apache setup is complete make sure that your webserver is reachable over https from the public internet. And also export the whole public certificate chain to the CMT to share your public key information with your e-Codex network partners. This is very important if you are using a not well trusted certificate authority or a self signed certificate.

Your shared webserver certificate will become part of the trusted_webserver.jks keystore - and only certificates within this store are considered trusted in the specific e-Codex Environment.

### Activating TLS-client-authentication/2-way-SSL on apache

Some e-Codex usecases may require also TLS-client authentication (aka 2-way-SSL). If the client authentication is used also the client is authenticated at the server on TLS level. For this purpose the client presents a client certificate to the webserver when opening the connection.

For activating 2-way-ssl on apache the following example configuration can be used. For further details consult the apache manual!

```
<VirtualHost _default_:443>
  SSLEngine on
  ServerName localhost:443
  SSLCertificateFile "${SRVROOT}/conf/ssl/server.crt"
  SSLCertificateKeyFile "${SRVROOT}/conf/ssl/server.key"
  # The following Parameter holds all trusted client certificates (2)
  # (and thier parent certificates) in pem-format.
  SSLCACertificateFile "${SRVROOT}/conf/ssl/trusted_clients.pem"
  DocumentRoot "${SRVROOT}/htdocs"

        CustomLog "${SRVROOT}/logs/ssl_request.log %t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x
        <Directory "${SRVROOT}/htdocs">
                Options Indexes Includes FollowSymLinks
                AllowOverride AuthConfig Limit FileInfo
        Require all granted
        </Directory>

  #always protect the whole v-host to avoid post-handshake authentication!
  SSLVerifyClient Require
  SSLVerifyDepth 2 # clientcert(0)->intermediateCA(1)->RootCA(2)

  <Location "/domibus/services/msh" >
    ProxyPass <gateway>/domibus/services/msh
    ProxyPassReverse <gateway>/domibus/services/msh
  </Location>

</virtualhost>
```

SSLVerifyDepth https://httpd.apache.org/docs/2.4/en/mod/mod_ssl.html#sslverifydepth

| | |
|---|---|
| Note | Put the SSLVerifyClient Require directive into the vhost directive to avoid SSL renegotiation. Though apache supports the SSLVerifyClient also within a Location directive but this would force the client to renegotiate the TLS connection. This can lead to connection errors at the as webclient acting domibus gateway. |

### Additional Notes on Webserver Setup

- Ensure TLS 1.3 is activated and downgrading is not allowed!
- Ensure that if the usecase requires it TLS-client authentication is activated

## AS4 Gateway / Domibus Gateway

Most of the e-Codex installations are using the domibus gateway for fullfilling the AS4 interface. But there are also known installations which are using IBM axway.

The eDelivery site of CEF lists some more alternatives: https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eDelivery+AS4+conformant+solutions

Keep in mind that if you are using some of the alternatives you still have to fullfill the e-Codex protocol demands. (TODO: link to technical differences ecodex - eDelivery)

### Example Setup

For installing the domibus gateway the website provides a very good guide. https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Domibus

### TLS-client-authentication/2-way-SSL authentication

For clientauthentication the information is located under:

<domibus_tomcat>/conf/domibus/clientauthentication.xml

```
<http-conf:tlsClientParameters disableCNCheck="true" secureSocketProtocol="TLSv1.2"
  xmlns:http-conf="http://cxf.apache.org/transports/http/configuration"
  xmlns:security="http://cxf.apache.org/configuration/security">

  <security:trustManagers>
    <!-- the following line contains the store which holds all trusted
    webserver certificates (file (b)) and thier parent certificates -->
    <security:keyStore type="JKS"
        password="changeit"
        file="${domibus.config.location}/keystores/trusted_https_servers.jks"
    />
  </security:trustManagers>
  <security:keyManagers keyPassword="12345">
  <!-- this key store contains the web client private key,
  the public certificate is distributed with
  the trusted_https_clients store  -->
    <security:keyStore
        type="JKS"
        password="12345"
        file="${domibus.config.location}/keystores/https_client_keystore.jks"
    />
  </security:keyManagers>
</http-conf:tlsClientParameters>
```

The same file is also used to list the trusted webservers the domibus gateway is connecting to. The certifcates (and thier intermediate and RootCAs) of the trusted webservers are listed within the trusted_https_servers.jks truststore.

To activate clientauthentication the gateway must present the webserver a key. This key is stored in the https_client_keystore.jks. This key is private! The corresponding certificate is distributed within the trusted_webclients.pem file and used at the webserver to allow the connection.

## Setting up Connector

The additional features of the e-Codex protocol like

- evidence handling
- ASIC-S container building
- TrustOK Token generatation

are provided by the domibusConnector. The domibus connector comes with an installation manual.

(TODO: set link to connector manual!)

# Configuration Update

The e-Codex environment establishes the connection betweent the access points (usually domibus gateway) over https connections. For this purpose multiple configuration parameters are necessary.

Table 1. Table Configuration Parameters

| Configuration Item | Description |
|---|---|
| Access-Point URL | The https address where the Gateway is reachable. <br> eg: https://ecodexgw.example.com/domibus/services/msh |
| Incoming IP-Addresses | IP-Addresses which are used for incoming traffic. This would enable your e-Codex partners to only allow traffic to this IP-Adresses |
| Outgoing IP-Addresses | IP-Addresses which the connection is made from. This will help your partners to only allow this specific ips to access thier webserver. Its mostly for a security reason to block unwanted access |
| TLS/SSL Webserver-Server certificate (1) | As the Request are sent over HTTPS the webserver needs a Certificate. An alias name of the certificate must match the domain name. The whole certificate-chain must be shared with your e-Codex partners. The certificate is used by the server to authenticate itself. The webserver client (the gateway, when it is sending a message) is validating this certificate. |
| TLS/SSL Webserver-Client certificate (2) | Not only the server needs to authenticate itself, also the client is required to authenticate itself at the webserver |
| Access-Point/Gateway Certificate (3) | The by the gateway sent messages are being signed. For this purpose the gateway needs a certificate. This certficiate must be distributed to the other partners to ensure trust and also to enable the other gateways to encrypt the message, when sent to your gateway. |
| Connector/e-Codex Certificate (4) | The by the connector generated ASIC-S Container and evidences are also signed. For this purpose the connector also needs a certificate. This certificate must also be shared with your partners to ensure trust |

All these configuration parameters can be configured on the CMT. The CMT is able to merge all this configurations together and generate a package for each e-Codex member. After this step the package must be downloaded and the configuration of each member has to be updated.

## Example Configuration Package

Example Configuration Package

Table 2. Table Distributed Configuration Files

| Name | Description |
|---|---|
| trusted_https_servers.pem (a) | The certificate chains of all trusted webservers (1) in the pem-format. |
| trusted_https_servers.jks (b) | The certificate chains fo all trusted webserver (1) in java key store format. |
| trusted_https_clients.pem (c) | The certificate chains of all trusted webserver (2) clients in pem-format. |
| trusted_https_clients.jks (d) | The certificate chains of all trusted webserver (2) clients in jks-format |
| trusted_gateways.jks (e) | The certificate chains of all trusted gateways (3) |
| trusted_connectors.jks (f) | The certificate chains of all trusted connectors (4) |
| eDeliveryAS4Policy.xml (g) | This file contains the ws-security-policy and defines which encryption and signature algorithms are allowed between 2 access-points. It also defines if 2-way TLS/SSL encryption/authentication is required or optional. |
| domibus-configuration-<country>.xml (h) | This file contains the p-Mode configuration. The p-modes are defining the connection parameters between two access-points. Within the p-Modes also the securityPolicy is referenced. They also define which message types are allowed. |

## JKS vs. PEM-Format

The jks format is the default keystore format in java. The PEM-Format on the other side is used within apache webservers. It contains the certificates Base64 encoded in a text file.

## Configuration Update Overview

The following files needs to be copied or uploaded:

- trusted_https_clients.pem (c) must be copied to the Webserver
- trusted_gateways.jks (e) must be uploaded via the domibus UI into the domibus gw
- domibus-configuration-<country>.xml (h) must be uploaded via the domibus UI into the domibus gw
- trusted_https_servers.jks (b) must be copied into the domibus conf keystores folder
- trusted_connectors.jks (f) must be copied into the domibus connector keystores folder
- domibus-configuration-<country>.xml (h) **must also** be uploaded via the admin UI into the domibus **connector**

Configuration update overview

## Update Webserver Configuration

If 2way-ssl is enabled the webserver is doing the client authentication. So the webserver configuration needs to be updated. If you are using the example configuration, the trusted_https_clients.pem file must be copied into the apache configuration. The following code-block contains an **example** Apache 2way-SSL-Configration.

```
<VirtualHost _default_:443>
  SSLEngine on
  ServerName localhost:443
  SSLCertificateFile "${SRVROOT}/conf/ssl/server.crt"
  SSLCertificateKeyFile "${SRVROOT}/conf/ssl/server.key"
  # The following Parameter holds all trusted client certificates (2)
  # (and thier parent certificates) in pem-format.
  SSLCACertificateFile "${SRVROOT}/conf/ssl/trusted_clients.pem"
  DocumentRoot "${SRVROOT}/htdocs"

        CustomLog "${SRVROOT}/logs/ssl_request.log" \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
        <Directory "${SRVROOT}/htdocs">
                Options Indexes Includes FollowSymLinks
                AllowOverride AuthConfig Limit FileInfo
    Require all granted
        </Directory>

  #always protect the whole v-host to avoid post-handshake authentication!
  SSLVerifyClient Require


  <Location "/domibus/services/msh" >
    ProxyPass <gateway>/domibus/services/msh
    ProxyPassReverse <gateway>/domibus/services/msh
  </Location>


</virtualhost>
```

## Update Gateway Configuration

The gateway needs to be updated on 3 places:

- The trusted webserver certificates (file (b))
- The trusted gateway certificates (file (e))
- The p-Modes itself (file (h))

Sometimes also an update of the security policy may be necessary.

Important This description only gives a brief overview. For more details consult the gateway administration guide!

### Update Gateway Client Configuration | trusted_https_servers.jks (b)

If a webserver certificate changes the trusted_https_servers.jks must be copied to the location configured within the clientauthentication.xml file. The file is usually located under

<domibus_tomcat>/conf/domibus/clientauthentication.xml

```
<http-conf:tlsClientParameters disableCNCheck="true" secureSocketProtocol="TLSv1.2"
  xmlns:http-conf="http://cxf.apache.org/transports/http/configuration"
  xmlns:security="http://cxf.apache.org/configuration/security">

  <security:trustManagers>
    <!-- the following line contains the store which holds all trusted
    webserver certificates (file (b)) and thier parent certificates -->
    <security:keyStore type="JKS"
        password="changeit"
        file="${domibus.config.location}/keystores/trusted_https_servers.jks"
    />
  </security:trustManagers>
  <security:keyManagers keyPassword="12345">
  <!-- this key store contains the web client private key,
  the public certificate is distributed with
  the trusted_https_clients store  -->
    <security:keyStore
        type="JKS"
        password="12345"
        file="${domibus.config.location}/keystores/https_client_keystore.jks"
    />
  </security:keyManagers>
</http-conf:tlsClientParameters>
```

### Update Trusted Gateways Configuration

The gateway certificates are updated by uploading the trusted_gateways.jks (file (e)) into the domibus gateway. This can be done with the admin ui. A restart is not required.

Figure 1. Upload Truststore

### Update p-Modes

The p-Modes can also be updated by uploading the new p-modes file with the admin-ui. A restart is not required.

## Update Connector Configuration

Within the connector the following parameters must be updated:

- p-Modes (connector: 4.0, 4.1)
- trusted connector certificates

## Update Connector p-Modes Configuration

The connector (4.0, 4.1) also requires the upload of the p-Modes. This can be done with the connector admin ui. Log into the admin ui and upload the p-Modes file.

## Update Trusted Connectors Certificates

Currently it is not possible to update the trusted-connector certificates within the admin-ui. The keystore has to be replaced. To do this copy the file (f) to

```
<connector_tomcat>/conf/connector/keystores/trusted_connectors.jks
```

And make sure that the connector property [connector.security.trust-store.path] points to that location! The following block contains an excerpt of the connector configuration with the relevant properties.

```
#############################  connector truststore  #############################

# Holds all public keys of partners to validate
# sent and received ASIC-S containers against.
#

connector.security.trust-store.path=file:./conf/connector/keystores/trusted_connectors.j
connector.security.trust-store.password=changeit
```

# Differences between e-Codex and e-Delivery

To be disappointing at the first written line. There are no differences! E-delivery is a part of e-Codex. E-Codex is using e-Delivery to transport the e-Codex messages between two access points.

## The features of e-Delivery

- Utilizes AS4 messaging standard
- Ensures message transport between two access-points
- Validation and compression of the user message
- Signing of the compressed message
- Encryption of the signed compressed message
- validation of the origin user message
- sending back of acknowledgments

Figure 1. Excerpt of the e-Delivery Flyer

https://ec.europa.eu/cefdigital/wiki/download/attachments/82773424/20180912_eDelivery_Flyer.pdf?version=1&modificationDate=1537358787680&api=v2

## The requirements of e-Codex

E-Delivery does a lot of things and is put to good use in a lot of projects. But for e-Codex there have been additional requirements.

- Proof of the location of the document
- Creating trust between different systems

## Proof of the location of the document

Within the e-Codex usescases there is always a urgent need to determine the current location of a sent document. Though the AS4 acknowledge tells us when the message has reached the other access-point, but for e-Codex there is also the requirement to tell exactly when the message has reached the inbox of the final recipient or for some usecaes when the final recipient has read/become aware of the message.

For this purpose e-Codex is generating ETSI-Rem evidences if messages have reached a specific point.

Usually within e-Codex the message is passed to an national system. Often this national systems have similar transport state messages. This transport state messages are mapped into ETSI-REM evidences and transported to the sending party. Back at the sending party this evidences can be ignored or mapped back to the national evidences.

## Creating trust between different systems

E-Codex was build to connect different message transport systems. Within this transport systems there are also different ways to ensure the trust/origin of the document. Some of this systems are using an PKI approach were all generated documents are signed by the creator. Within other environments the document hashes are stored in a central database. The validity is checked by the database. Within other environments the necessary documents are not really created, they are just values and objects within an application.

All these examples combine the problem that documents which are exported from this environments cannot be validated within an other environment.

To solve this problem within e-Codex the idea of the trust token was created. The basic concept is before any documents are sent to another domain the sending instance is creating an signed ASIC-S container. This ASIC-S container also contains a token within this token the trust state of the sent document for the current environment ist documented (as XML for automatic processing and as PDF for human readability).

After this steps the document is transported to the other domain. Before the document is importated in this domain the receiving side is checking the ASIC-S container and the trust token. First of all the ASIC-S container is validated. In any case the ASIC-S container has to be valid within the e-Codex environment. After that the trust ok token is being checked. And its completly the decision of the receiving domain if and how the document is imported into the receiving system.