Submitted to the EC on 13/09/2014

COMPETITIVENESS AND INNOVATION FRAMEWORK PROGRAMME
ICT Policy Support Programme (ICT PSP)

e-CODEX

e-Justice Communication via Online Data Exchange

ICT PSP call identifier: CIP-ICT-PSP-2009-4

ICT PSP main Theme identifier: CIP ICT PSP 2010 5.2 3: E-JUSTICE SERVICES

Project full title: e-Justice Communication via Online Data Exchange

Grant agreement n°: 270968

D 4.9: Developed Modules and Building Blocks (Update of D4.3)

Deliverable Id :	D 4.9
Deliverable Name :	Developed Modules and Building Blocks
Status :	V 1.0
Dissemination Level :	PU
Due date of deliverable :	31.07.2013
Actual submission date :	13.09.2014
Work Package :	WP4
Organisation name of lead partner for this deliverable :	Ministry of Justice, Estonia
Author (s):	Rudi Teschner Adrian Klar Lesli Hommik
Partner (s) contributing :	DE, EE, WP1, WP4, WP5, WP7

Abstract:

As part of the e-CODEX project, this deliverable provides the information about developed modules and building blocks of WP4 concentrating on the different functionalities of the e-CODEX Trust Library and especially on the Trust OK-Token.

List of Figures

List of Tables

History

Version	*Date*	*Changes made*	*Modified by*
0.4	08.07.2014	Update of Deliverable D4.3	Adrian Klar
0.5	28.07.2014	Additional updates of Deliverable D4.3	Lesli Hommik
0.8	26.08.2014	Changes made based on the comments from the first review cycle.	Adrian Klar
0.9	11.09.2014	Finalization	Lesli Hommik
1.0	13.09.2014	Final editorial amendments	WP1

List of Abbreviations and Acronyms

Acronym	*Explanation*
Advanced Electronic System	System that fulfils the requirements defined in D4.2, page 16
API	Application Programming Interface
ASiC	Associated Signature Container, published by ETSI as TS 102 918
ASiC-S	Simple form of ASiC
Binary Files	Encoded file that contains any type of data
Bouncy Castle	Collection of APIs used in cryptography
CAdES	CMS Advanced Electronic Signatures, published by ETSI as TS 101 733
CMS	Cryptographic Message Syntax
Connector Framework	Generic connector developed by WP5 where the e-CODEX Trust Library is integrated
CRL	Certificate Revocation List, see "RFC 5280" http://www.ietf.org/rfc/rfc5280.txt
DG	Directorate-General
DG MARKT	DG Internal Market and Services
DSS	Digital Signature Services: open source signing and validation library
e-CODEX	e-Justice Communication via Online Data Exchange
ETSI	European Telecommunications Standards Institute
Factory Method Pattern	Object-oriented creational design pattern to deal with the problem of creating objects without specifying the exact class of that object
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
Input Stream	Sequence of data
Interface	Point of interaction between systems
Java	General-purpose, class-based and object-oriented programming language
Maven	Build automation tool mainly used for Java projects
Method Chaining	Common technique for invoking multiple method calls
Model View Controller Principle	Architectural pattern used in software engineering

OCSP	Online Certificate Status Protocol, see “RFC 2560” http://www.ietf.org/rfc/rfc2560.txt
Open source	Methodology that promotes free redistribution and access
PAdES	PDF Advanced Electronic Signature, published by ETSI as TS 102 778
PDF	Portable Document Format
PEPPOL	Pan-European Public Procurement Online http://www.peppol.eu/
Proxy	Computer network service to create connections to other network services
Service Provider	e-CODEX Service Provider provides services to users or general services under the responsibility of a public authority
SPOCS	Simple Procedures Online for Cross- Border Services http://www.eu-spocs.eu/
Spring	Open source application framework for the Java platform
STORK	Secure Identity across borders linked https://www.eID-stork.eu/
Thread Safety	Computer programming concept that guarantees safe multiple-thread access to shared data structures at the same time
Trust OK-Token	Token that provides the possibility for a receiving party to recognize documents filed by using a trustworthy advanced electronic system based on either signature or authentication
TSL	Trust-Service Status List, published by ETSI as TS 102 231
WP	Work Package
XAdES	XML Advanced Electronic Signature, published by ETSI as 101 903
XML	eXtensible Markup Language
ZIP	File format for data compression and / or archiving

Executive Summary

The goal of e-CODEX is to improve cross-border access for citizens and businesses to legal means in Europe and the interoperability between legal authorities by using instruments of the ICT.

WP4 aims to cover all e-Identity and e-Signature related topics:

- e-Identity management for natural and legal roles, mandates and rights as well as user authentication and authorisation
- Verification and Implementation of e-Signatures.

The present document is the updated version of the fourth deliverable written by WP4. It describes the implementation of the conceptual deliverable D4.2/D4.8 that is based on the analysis in the deliverables D4.1 and D4.1.1.

The focus is on the library e-CODEX Connector-Container Services (e-CODEX Trust Library) developed by ARHS along with a corresponding system documentation. This deliverable is an extension of the documentation DSS4eCodex-SD-System Documentation-v1.08.doc.

This description of the library includes information about its functionalities, structure and usage. In addition to providing a general overview, it describes the functionalities that have been implemented according to the requirements in D4.2 and the ARHS contract. It also considers the structure of the library, the separation of the classes into several packages influenced by the chosen Model-View-Controller software architecture, the class descriptions and the public methods.

Since the last version of this deliverable has been published, the library has progressed and has been subject to further developments. These developments include additional new functionalities like the solution for handling authentication-based systems where the service provider's signs the issued documents, as well as the handling and elimination of detected problems and the upgrade to DSS Version 4[1] released in June 2014.

The changes made need to be documented properly to provide an accurate basis for future usage of the library. This is vital as the extension of the project involves several new partners.

This deliverable contains information about the configuration and integration of the e-CODEX Trust Library and explains how to use it either as part of the connector framework developed by WP5 or as part of an individual solution.

With this deliverable, the system documentation and the provided test classes the developers in each member state will be able to understand and use the library.

[1] Digital Signature Service, https://joinup.ec.europa.eu/asset/sd-dss/description

1 Introduction

1.1 Scope and Objective of Deliverable

This deliverable provides the information about developed modules and building blocks of WP4, concentrating on the different functionalities of the e-CODEX Trust Library. It describes its overview, workflow, functionalities and architecture and also its utilisation by further elaborating the configuration and implementation process.

1.2 WP4 General Objectives and Vision

The main objective for WP4 is to deal with electronic identity and electronic signatures. Due to the nature of the e-CODEX pilot use cases, WP4 concentrates on electronic signatures. Providing a solution for handling electronic signatures is essential for a successful piloting phase as signatures are especially crucial in the field of justice.

1.3 Methodology of Work

The deliverable was drafted by WP4 author team which consists of IT-architects from Germany under the supervision of the WP leader from Estonia. The software from ARHS and its system documentation as well as the developments made within WP4 internal team were analysed and used as a basis for this deliverable. Team members from Turkey and Poland were included in the testing of the software and their feedback was taken into account. Additional tests on the final version of the library were done by an external tester from Germany to ensure objective test results.

1.4 Relations to Internal e-CODEX Environment

This deliverable is important for the piloting Member States to give them an understanding about the e-CODEX Trust Library and its functionalities to help them to utilise the DSS tool.

1.5 Relations to External e-CODEX Environment

The aim is to provide a description of modules and building blocks that have been realised, concentrating on the different functionalities of the e-CODEX Trust Library. In deliverable D4.2/D4.8, the modules and building blocks were described in detail and solutions were specified. Developments of the software code are based on the requirements written down in deliverable D4.2/D4.8. This deliverable will have a great impact on the piloting since it contains information about the configuration and implementation of the software code.

1.6 Quality Management

External quality checks have been performed by the External Quality Manager. Internal quality checks have been done by WP1 team as well as the members of WP4.
The following table gives an overview about the quality checks performed on this deliverable.

Category	Remarks	Checked by
Conformance to e-Codex template	Firstly done by WP4 leader and also checked by WP1 before submission to EC.	WP4 WP1 EQM
Language & Spelling	Remarks from EQM were taken into account and the deliverable was re-checked by WP4 leader before submission.	WP4 EQM
Delivered on time		
Each technology description contains the correct elements	Checked by IT-architects working on the deliverable.	WP4
Consistency with description in the TA and in other e-Codex deliverables	Checked by WP4 leader and WP1.	WP4 WP1
Content is fit for purpose	Checked by IT-architects working on the deliverable.	WP4
Content is fit for use	Checked by IT-architects working on the deliverable.	WP4
Commitment within WP	Checked by WP4 leader.	WP4

Table 1: Quality Checklist

1.7 Risk Management

The following table gives an overview of the main risks of WP4:

Description	Probability	Impact	Priority	Response	Owner
Partners not contributing in WP4 which causes delays in deliveries.	High	High	Very high	Involvement of new partners in WP4 and enforcement and encouragement of contribution by WP4 leader and the coordinator.	WP4
Problems in making the solution work for every piloting country	High	Medium	High	Close collaboration with piloting countries, including piloting countries in testing and close collaboration with the developer.	WP4
Problems in integrating the WP4 library into the Connector	High	Low	Medium	Close collaboration with other WP-s, piloting countries and developers in order to make sure that integration is successful.	WP4
National solutions are not in accordance with the standards and regulations and can't be integrated into the developed solution.	Medium	Medium	Medium	Member States have to modify their national solutions to be in accordance with given standards and regulations.	WP4
We are unable to create a working solution for e-CODEX.	Medium	High	High	Experts and good developers need to be included in the specification and development phase.	WP4

Table 2: Risks

1.8 Structure of the Document

The document is structured as follows:

1. Introduction
2. e-CODEX Trust Library
   2.1 Overview
   2.2 Workflow
   2.3. Functionality
      2.3.1 Creation of a Trust OK-Token
      2.3.2 Creation of a signed ASiC-S container
      2.3.3 Verification of an ASiC-S container
      2.3.4 Reception of an ASiC-S container
      2.3.5 Application of an additional signature to an ASiC-S container
      2.3.6 Application of National Signature Settings
      2.3.7 Application of National Truststore Settings
      2.3.8 Application of National Validation Settings
      2.3.9 Proxy Configuration
   2.4 Architecture
3. Library Utilisation
   3.1 Configuration
   3.2 Implementation
4. References
5. APPENDIX: Basic Library Documentation

2 e-CODEX Trust Library

2.1 Overview

This chapter describes the library that ~~will be~~ is integrated into the e-CODEX connector framework. It provides the functionality to handle signatures and enables the connector to issue a Trust OK-Token based on the "Circle of Trust[2]" which preserves the relation between the documents by using an ASiC-S container.

The Trust OK-Token evaluates the integrity of the corresponding business document and its trust level. With the business document originating from either an authentication-based advanced electronic system or a signature-based advanced electronic system, the token verifies the source of the document and / or the signature that is applied on it.

The specific requirements have been defined in deliverable D4.8 and the "ARHS contract[3]".

Name of the Library: ecodex-container-1.8.jar[4]

The library provides the following main functionalities:

- Creation of a Trust OK-Token
- Creation of a signed ASiC-S container
- Verification of the ASiC-S container
- Reception of an ASiC-S container
- Application of an additional signature to an ASiC-S container

In addition to the basic functionality, it also covers configurational aspects:

- Application of National Signature Policies
- Application of National Validation Policies
- Proxy Configuration

All functionalities are described in detail in chapter 2.3.

---

[2] Defined in "Circle of Trust Agreement", not published yet

[3] ARHS e-CODEX Connector Signing & Validation Solution Proposal, contract not published

[4] Version date: 25.06.2014

2.2 Workflow

Being part of a national connector, or more precisely the connector framework, the library serves different purposes depending on whether the national connector is sender or recipient of the message.

Sending side:

1. Receive the (possibly signed) business document and its attachments
2. Check the business document against the configured technical and legal validation services
3. Generate and sign the Trust OK-Token
4. Create the content archive and wrap up all documents in an ASiC-S container
5. Sign the container to ensure data integrity

Receiving side:

1. Receive the signed ASiC-S container
2. Check the signatures on the container and the tokens using the DSS validation service
3. Extract the documents from the container if necessary
4. Provide the documents

For a visualisation of these steps, please see Figure 1: Basic Workflow on the next page.

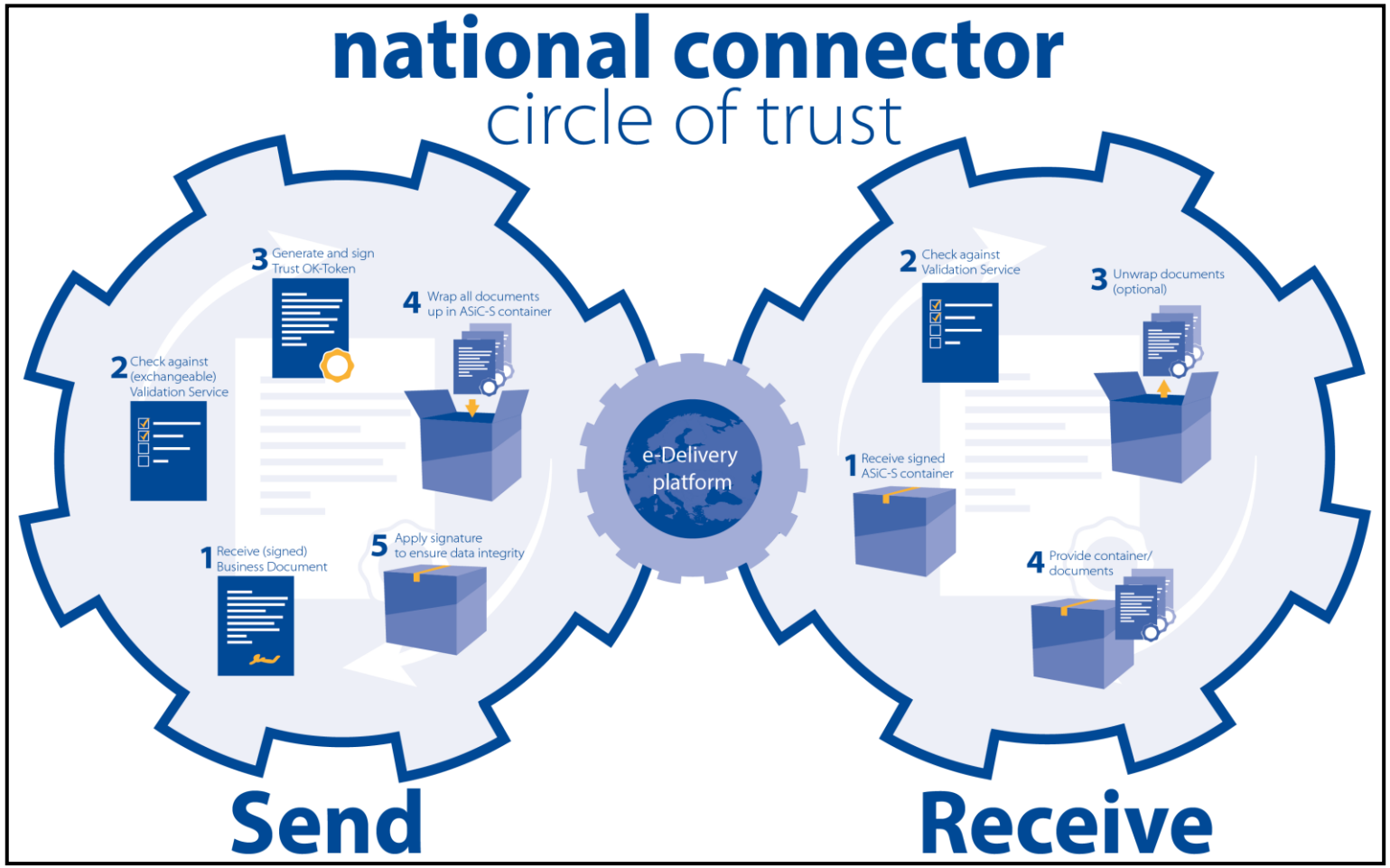

Figure 1: Basic Workflow

2.3 Functionality

2.3.1 Creation of a Trust OK-Token

The main purpose of the library is to provide means to create a Trust OK-Token and ensure its integrity. To do so, the target business document, its attachments and information about the token issuer have to be known in the connector framework. The information about the token issuer includes the name of the service provider, the country information as well as the type of advanced electronic system of the document source.

The service performs a technical and a legal validation of the business document and its results will be part of the information represented in the Trust OK-Token. The extent of the validation depends on the type of advanced electronic system. While the validation for authentication-based systems needs to be addressed specifically, the delivered connector framework covers the validation of documents issued with signatures via either signature-based advanced electronic systems or authentication-based advanced electronic systems with signature. Through an analysis of the signatures applied to the business document using the DSS[5] tool, the technical trust level is evaluated. The legal result then depends on the used system: In case of a signature based advanced electronic system the legal result is derived solely from the technical result. An authentication-based advanced electronic system with signature also takes the validation of the signature certificate against a TSL of trusted authentication service providers into account.

The Trust OK-Token will be provided both as human-readable PDF and as machine-readable XML and will be signed according to the configuration of the connector framework. This process implies the creation of an ASiC-S container according to chapter 2.3.2.

This process can be initialised independently by calling the following public method within an implementation of the interface ECodexContainerService.java:

```
ECodexContainer create(BusinessContent businessContent, TokenIssuer issuer)
```

This method is used in the delivered connector framework.

[5] Digital Signature Service, https://joinup.ec.europa.eu/asset/sd-dss/description

2.3.2 Creation of a signed ASiC-S container

The business document with its attachments and the generated PDF Trust OK-Token that includes the validation report will be placed inside the ASiC-S container whereas both XML versions, the business document and the Trust OK-Token, will remain outside.

This method is used in the delivered connector framework being part of the creation of a Trust OK-Token and cannot be addressed directly.

2.3.3 Verification of an ASiC-S container

The receiving connector needs to be able to validate the consistency and reliability of the container. This will be done by checking the signature applied to the container, but also includes a validation of both versions of the Trust OK-Token.

It verifies if the signature applied to the ASiC-S container is valid, that the XML token contains the signature and that the PDF token is signed properly.

Problems will be reported back to the connector framework which needs to be configured to decide how to proceed ~~than~~ afterwards.

Additionally, the sending connector is also able to use this functionality to check a created container before submission to prevent the risk of sending defective containers.

This process can be initialised independently by calling the following public method within an implementation of the interface ECodexContainerService.java:

```
CheckResult check(ECodexContainer container)
```

This method is used in the delivered connector framework being a component of the reception of an ASiC-S container.

2.3.4 Reception of an ASiC-S container

The connector framework enables the receiving connector to run a service that receives and processes an input stream which represents the ASiC-S container and the XML version of the Trust OK-Token. The service is able to unmarshal the data to its entities and creates an ASiC-S container object holding this information. This method does not include verification of the ASiC-S container by default, but the delivered connector framework covers this aspect by addressing the functionality described in the previous chapter: Verification of an ASiC-S container.

This process can be initialised independently by calling the following public method within an implementation of the interface ECodexContainerService.java:

```
ECodexContainer receive(InputStream asicInputStream, InputStream tokenStream)
```

This method is used in the delivered connector framework.

2.3.5 Application of an additional signature to an ASiC-S container

The receiving connector is able to apply its own signature to a received ASiC-S container using the connector framework configuration. Neither the business document and its attachments nor the previously applied signature on the container by the sending connector will be altered. But it will give the receiving connector the ability to increase the acceptance of the Trust OK-Token by providing a kind of “signature chain” for the end user.

The sending connector is also able to use this functionality, but there is no benefit from doing so.

This process can be initialised by calling the following public method within an implementation of the interface ECodexContainerService.java:

```
ECodexContainer addSignature(ECodexContainer container)
```

This method is provided in the delivered connector framework and can be used in specific implementations if necessary.

2.3.6 Application of National Signature Settings

The library enables modification of signature settings that are required to sign the Trust OK-token and the ASiC-S container including information about the used certificate as well as algorithms.

Figure 2 shows the example that is provided in chapter 8.2 of the system documentation.

```
private SignatureParameters createSignatureParameters() throws Exception {
    final InputStream pkcs12Input = getClass().getResourceAsStream("/certificates/server_signature.p12");
    final byte[] pkcs12Data = IOUtils.toByteArray(pkcs12Input);
    final RFC3370Pkcs12SignatureToken p12SignatureToken = new RFC3370Pkcs12SignatureToken("password".toCharArray(), pkcs12Data);
    final KSPrivateKeyEntry keyEntry = (KSPrivateKeyEntry) p12SignatureToken.getKeys().get(0);

    final SignatureParameters params = new SignatureParameters();

    params.setPrivateKey(keyEntry.getPrivateKey());
    params.setSignatureAlgorithm(keyEntry.getSignatureAlgorithm().getName());
    params.setCertificate(keyEntry.getCertificate());

    final List<X509Certificate> x509Certs = new ArrayList<>();
    final Certificate[] certificates = keyEntry.getCertificateChain();
    for (final Certificate certificate : certificates) {
        if (certificate instanceof X509Certificate) {
            x509Certs.add((X509Certificate) certificate);
        }
    }
    params.setCertificateChain(x509Certs.toArray(new X509Certificate[x509Certs.size()]));
    params.setDigestAlgorithm(DigestAlgorithm.SHA1.getName());

    return params;
}
```

Figure 2: createSignatureParameters()

The connector framework uses Java Spring technology which enables to configure the different parameters in a configuration file. So the settings, in particular the certificate information, need to be adjusted for each connector using the file connector.properties.

Configuration Parameters:

- connector.security.keystore.path
- connector.security.keystore.password
- connector.security.key.alias
- connector.security.key.password

```
<bean id="securityContainer" class="eu.ecodex.connector.security.container.ECodexSecurityContainer">
  <property name="javaKeyStorePath" value="${connector.security.keystore.path}" />
  <property name="javaKeyStorePassword" value="${connector.security.keystore.password}" />
  <property name="keyAlias" value="${connector.security.key.alias}" />
  <property name="keyPassword" value="${connector.security.key.password}" />
  <property name="containerService" ref="containerService"/>
  <property name="tokenIssuer" ref="tokenIssuer"/>
</bean>
```

Figure 3: Excerpt from ECodexConnectorSecurityToolkitContext.xml (Signature Settings)

2.3.7 Application of National Truststore Settings

The certificates of every trusted connector can be listed within a JKS truststore. This truststore can be configured using the following settings of the file connector.properties.

- java.truststore.path
- java.truststore.password

The verification of the certificate against the configured truststore is realized ad the time of container verification as described in chapter 2.3.3.

```
<bean id="trustedCertificates" class="eu.ecodex.dss.model.CertificateStoreInfo" >
    <property name="location" value="${java.truststore.path}" />
    <property name="password" value="${java.truststore.password}" />
</bean>
```

Figure 4: Excerpt from ECodexConnectorSecurityToolkitContext.xml (Truststore Settings)

2.3.8 Application of National Validation Settings

The connector is able to use specific implementations for the technical and legal validation instead of the provided validation services. To do so, these services are designed to be exchangeable. Interfaces with the required method declarations have been created to provide a basis for adaptions if needed.

By default, the technical validation is performed using the DSS tool. The result of the validation and the validation report will be integrated into the Trust OK-Token. ~~Figure 1~~Figure 5 shows an example for setting up a specific technical validation service and can be found in the ExampleNSPTest01.java[6].

[6] These files can be found in the ecodex-container-1.8.jar

```
@Override
protected ECodexTechnicalValidationService createTechnicalValidationService() {
    tecValServiceImpl = new AbstractNSPTechnicalValidationService() {
        @Override
        protected void init() {
            issuer = new TokenIssuer();
            issuer.setCountry("DE");
            issuer.setServiceProvider("A German specific solution");
            issuer.setAdvancedElectronicSystem(AdvancedSystemType.AUTHENTICATION_BASED);

            valResult_TrustLevel = TechnicalTrustLevel.SUCCESSFUL;
            valResult_Comments = "green";

            org.w3c.dom.Document diagnosticDataDocument = DSSXMLUtils.buildDOM(getClass().getResourceAsStream("/diagnostic-data.xml"));
            final DiagnosticData diagnosticData = new DiagnosticData(diagnosticDataDocument);
            org.w3c.dom.Document simpleReportDocument = DSSXMLUtils.buildDOM(getClass().getResourceAsStream("/simple-report.xml"));
            final SimpleReport simpleReport = new SimpleReport(simpleReportDocument);
            valReport_Data = Arrays.asList(diagnosticData, simpleReport);

            authInfo_IdentityProvider = "unknown-identity-provider";
            authInfo_UsernameSynonym = "unknown-user";
            authInfo_Time = createXMLGregorianCalendar(null);
        }

        @Override
        public Document createReportPDF(final Token token) throws ECodexException {
            if (token.getIssuer().getAdvancedElectronicSystem() == AdvancedSystemType.AUTHENTICATION_BASED) {
                return null;
            } else {
                return super.createReportPDF(token);
            }
        }
    };

    return tecValServiceImpl;
}
```

Figure 5: Exemplary Set-Up of a specific Technical Validation Service

The default legal validation service of the delivered library provides just a basic implementation of a legal evaluation. The result of the legal validation is based on the result of the technical validation: It is only successful if the technical validation is successful. This basic implementation can be found in the file DSSECodexLegalValidation.java[6] and more complex validation scheme can be easily applied.

2.3.9 Proxy Configuration

The library enables configuration of HTTP/HTTPS proxy settings. Internet access is required to validate a signature against its TSL and check whether the certificate has been revoked.

To adjust the proxy settings, the connector framework offers the possibility to set the values for the HTTP proxy configuration in the file connector.properties.

Configuration Parameters:

- http.proxy.enabled
- http.proxy.host
- http.proxy.port
- http.proxy.user
- http.proxy.password

Figure 6 lists a part of the configuration file to show the further processing of the required values.

```
<bean id="securityContainer" class="eu.ecodex.connector.security.container.ECodexSecurityContainer">
  <property name="javaKeyStorePath" value="${java.keystore.path}" />
  <property name="javaKeyStorePassword" value="${java.keystore.password}" />
  <property name="keyAlias" value="${key.alias}" />
  <property name="keyPassword" value="${key.password}" />
  <property name="containerService" ref="containerService"/>
  <property name="tokenIssuer" ref="tokenIssuer"/>
</bean>

<bean id="connectorProxyDao" class="eu.ecodex.connector.security.proxy.ECodexConnectorProxyDao">
  <constructor-arg value="${http.proxy.enabled}"/>
  <constructor-arg value="${http.proxy.host}"/>
  <constructor-arg value="${http.proxy.port}"/>
  <constructor-arg value="${http.proxy.user}"/>
  <constructor-arg value="${http.proxy.password}"/>
</bean>

<bean id="proxyPreferenceManager" class="eu.europa.ec.markt.dss.manager.ProxyPreferenceManager">
  <property name="proxyDao" ref="connectorProxyDao"/>
</bean>
```

Figure 6: Excerpt from ECodexConnectorSecurityToolkitContext.xml (Proxy Configuration)

2.4 Architecture

The architecture of the library is derived from the Model-View-Controller principle with the advantage that through separating the information from the actual services, these services are exchangeable and can be replaced by specific implementations to fulfil the national requirements and connect to the national solutions.

The models consist of application data and business rules whereas the controller defines the behaviour of the application and provides the interface to be used by the national system.

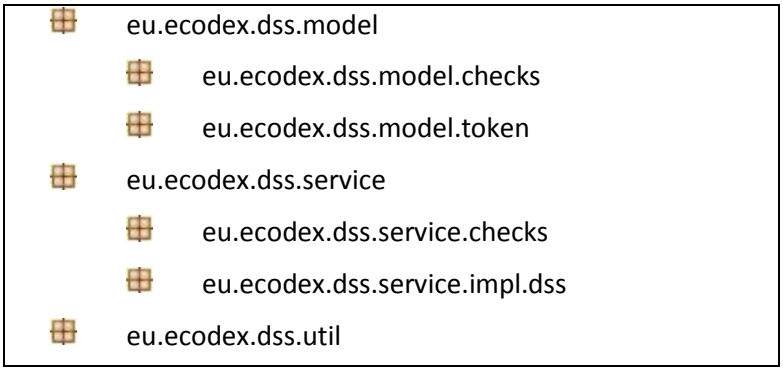

Figure 7: Package Overview

A complete overview of the packages, classes, interfaces and their methods can be found in the Appendix: Basic Library Documentation. An overview of the entities, their attributes and their relations can be found in the system documentation[7] of the library, page 15.

[7] DSS4eCodex-SD-System Documentation-v1.08.doc

3 Library Utilisation

3.1 Configuration

The library ecodex-container-1.8.jar that contains the binary files is delivered together with several other files. Along these, the following are the most important documents to start using the library:

- the maven project file ecodex-container-1.8-project.zip and
- the maven dependency libraries ecodex-container-1.8-mavenlibs.zip
- the documentation DSS4eCODEX-SD-System Documentation-v1.08.doc

To use the library, it is necessary to set up the configuration properly.

This includes:

- Proxy configuration (optional)

  Instantiate and apply ProxyPreferenceManager and EnvironmentConfiguration objects
- Security provider

  Add BouncyCastle as security provider
- Trust model

  Create TSL, OCSP and CRL sources and apply them to a TrustedListCertificateVerifier object
- Validation services

  Create DSS Validation services and apply verifier, proxy manager and environment configuration
- Signature parameters

  Create the connector signature and truststore settings
- Container service

  Instantiate the container service and apply the environment configuration, the signature parameters, the validation service and the verifier

Examples and more details can be found in the documentation[8] and the provided test classes that can be found in the subfolder /src/test/ of the e-CODEX Trust Library[9].

[8] DSS4eCodex-SD-System Documentation-v1.08.doc

[9] ecodex-container-1.8.jar

3.2 Implementation

The library provides services that implement the usage of the DSS tool for technical validation and a basic scheme for legal validation. These services are intended to be replaceable by specific implementations to match the requirements of national systems and reuse existing solutions.

To do so, interfaces have been created that can be used to implement existing validation services:

- ECodexTechnicalValidationService
- ECodexLegalValidationService
- ECodexContainerService

ECodexTechnicalValidationService

Defines the interfaces for the implementation of the technical validation service:

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

public TokenValidation create (Document document, Document detachedSignature)

public Document createReportPDF (Token token)

Example: DSSECodexTechnicalValidationService.java

ECodexLegalValidationService

Defines the interfaces for the implementation of the legal validation service:

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

public LegalValidationResult create (Token token)

Example: DSSECodexLegalValidationService.java

ECodexContainerService

Defines the interfaces for the implementation of the container service:

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

public void setContainerSignatureParameters (SignatureParameters signParams)

public void setTechnicalValidationService (ECodexTechnicalValidationService tvs)

public void setLegalValidationService (ECodexLegalValidationService lvs)

public ECodexContainer create (BusinessContent businessContent, TokenIssuer issuer)

public ECodexContainer receive (InputStream asicInputStream, InputStream tokenStream)

public CheckResult check (ECodexContainer container)

public ECodexContainer addSignature (ECodexContainer container)

Example: DSSECodexContainerService.java

Additionally, abstract classes for technical and legal validation services exist. They provide a very basic implementation to help construct new services. These classes are:

AbstractNSPLegalValidationService

protected abstract void init()

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

public LegalValidation Result create (Token token)

AbstractNSPTechnicalValidationService

protected abstract void init()

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

public TokenValidation create (Document businessDocument, Document detachedSignature)

protected TokenValidation _createValidation (Document businessDocument, Document detachedSignature)

protected TechnicalValidationResult _createValidation_1_ValidationResult ()

protected OriginalValidationReportContainer _createValidation_2_OriginalValidationReport ()

protected ValidationVerification _createValidation_3_Verification ()

protected ValidationVerification _createValidation_3_Verification_SignatureBased ()

protected ValidationVerification _createValidation_3_Verification_AuthenticationBased ()

public Document createReportPDF (Token token)

protected void _createPDF (Token token, com.lowagie.text.Document document)

public static XMLGregorianCalendar createXMLGregorianCalendar (Date date)

Depending on the scenario, the init () methods need to be modified to handle specific attributes or if the extending class has to overwrite some or all of the protected methods (underscore prefix).

4 References

ASiC-S TS 102918 v1.2.1	http://www.etsi.org/deliver/etsi_ts/102900_102999/102918/01.02.01_60/ts_102918v010201p.pdf
ASiC-S TS 103174 v2.1.1	http://www.etsi.org/deliver/etsi_ts/103100_103199/103174/02.01.01_60/ts_103174v020101p.pdf
XAdES TS 101903 v1.4.1	http://uri.etsi.org/01903/v1.4.1/ts_101903v010401p.pdf
PAdES TS 103172 v2.2.1	http://www.etsi.org/deliver/etsi_ts/103100_103199/103172/02.02.01_60/ts_103172v020201p.pdf
DSS 2.0 Digital Signature Service	https://joinup.ec.europa.eu/asset/sd-dss/description

5 Appendix: Basic Library Documentation

This chapter lists all classes, interfaces and enumerations. This includes information on each public and protected method, its return value and required parameters as well as a general description of its functionality.

Attributes are by default private but can be accessed by using provided getter and setter methods.

5.1 Package: eu.ecodex.dss.model

This package contains the basic entities with their attributes and their getter and setter methods.

Class: BusinessContent

Model class for the business document and its attachments

Methods:

public BusinessContent ()

Default constructor

public Document getDocument ()

Accesses the business document

public BusinessContent setDocument (Document document)

Sets the business document

public boolean hasDocument ()

Checks whether a business document is set

public Document getDetachedSignature ()

Accesses the optional detached signature for the business document

public BusinessContent setDetachedSignature (Document document)

Sets the optional detached signature for the business document

public boolean hasDetachedSignature ()

Checks whether a detached signature is set

public List<Document> getAttachments ()

Accesses the list of attached documents

public BusinessContent[10] setAttachments (List<Document> attachments)

Sets a list of documents as attachments

public BusinessContent addAttachment (Document attachment)

Adds a document to the list of attachments

public boolean hasAttachments ()

Checks whether at least one attachment is set

Class: CertificateStoreInfo

Class that contains information to access the certificate store holding the e-CODEX connector certificates that are required for the validation process

public String getLocation ()

Returns the URL for loading the keystore

public CertificateStoreInfo setLocation (String v)

Sets the keystore URL

public String getPassword ()

Returns the password for accessing the keystore

public CertificateStoreInfo setPassword (String v)

Sets the keystore password

public boolean isValid ()

Checks whether the URL is not empty

public String toString ()

Overrides the default toString-Method of this object to transport the information of the object in an easily readable way

[10] The principle to return the same object when calling methods that usually do not return values is called chaining. It enables the programmer to call multiple methods consecutively.

Interface: Document

Interface that enables access to the data of a document

The methods in this interface are just declarations. The actual logic needs to be addressed in the classes that implement this interface.

Methods:

public InputStream openStream ()

Provides the content of a document as InputStream

public String getName ()

Provides the name of a document

public MimeType getMimeType ()

Provides the MimeType of a document

Class: ECodexContainer

Model class that contains the signed content and the created ASiC document

Methods:

public ECodexContainer ()

Default constructor

public BusinessContent getBusinessContent ()

Provides the business content that is stored within the container

public ECodexContainer setBusinessContent (BusinessContent content)

Sets the Business Content

public Token getToken ()

Accesses the Trust OK-Token object structure stored within the container

public ECodexContainer setToken (Token token)

Sets the Trust OK-Token object structure

public Document getTokenXML ()

Returns the signed XML version of the Trust OK-Token

public ECodexContainer setTokenXML (Document tokenXML)

Sets the XML version of the Trust OK-Token

public Document getTokenPDF ()

Returns the signed PDF version of the Trust OK-Token

public ECodexContainer setTokenPDF (Document tokenPDF)

Sets the PDF version of the Trust OK-Token

public Document getAsicDocument ()

Returns the generated ASiC-S file

public ECodexContainer setAsicDocument (Document asicDocument)

Sets the ASiC-S file

public Document getBusinessDocument ()

Provides the business document that is stored within the container

public List<Document> getBusinessAttachments ()

Provides the attachments that are stored within the container

public Document getBusinessSignature ()

Provides the detached signature of the business document

Class: EnvironmentConfiguration

Model class for the configuration of the environment

Methods:

public ProxyData getProxyHTTP ()

Returns the proxy information for http connections

public EnvironmentConfiguration setProxyHTTP (ProxyData proxyHTTP)

Sets the http proxy information

public boolean isProxyHTTPValid ()

Checks whether the http proxy information is set and valid

public ProxyData getProxyHTTPS ()

Returns the proxy information for https connections

public EnvironmentConfiguration setProxyHTTPS (ProxyData proxyHTTPS)

Sets the https proxy information

public boolean isProxyHTTPSValid ()

Checks whether the https proxy information is set and valid

public CertificateStoreInfo getConnectorCertificates ()

Returns the information how to obtain the certificates of all e-CODEX connectors

public EnvironmentConfiguration setConnectorCertificates (CertificateStoreInfo v)

Sets the certificate store information

public boolean isConnectorCertificatesValid ()

Checks whether the store information for the e-CODEX connectors is valid

~~Subclass: **public static ProxyData**~~

~~*Class that contains the attributes to use a proxy connection*~~

~~public ProxyData setHost (String host)~~

~~Sets the host attribute required for a proxy connection~~

~~public String getHost ()~~

~~Returns the host attribute~~

~~public ProxyData setPort (Int port)~~

~~Sets the port attribute required for a proxy connection~~

~~public Int getPort ()~~

~~Returns the port attribute~~

~~public ProxyData setAuthName (String authenticationName)~~

~~Sets the authentication name~~

~~public String getAuthName ()~~

~~Returns the user attribute~~

~~public ProxyData setAuthPass (String authenticationPassword)~~

~~Sets the password attribute~~

~~public String getAuthPass ()~~

~~Returns the set authentication password~~

~~public boolean hasAuth ()~~

~~Checks whether authentication data (name & pass) is provided~~

public boolean isValid ()

Checks whether the host is not empty and the port is greater than 0

public String toString ()

Overrides the default toString-Method of this object to transport the information of the object in an easily readable way

Subclass: **public static CertificateStoreInfo –**

Class that contains information to access the certificate store holding the e-CODEX connector certificates that are required for the validation process

public String getLocation ()

Returns the URL for loading the keystore

public CertificateStoreInfo setLocation (String v)

Sets the keystore URL

public String getPassword ()

Returns the password for accessing the keystore

public CertificateStoreInfo setPassword (String v)

Sets the keystore password

public boolean isValid ()

Checks whether the URL is not empty

public String toString ()

Overrides the default toString-Method of this object to transport the information of the object in an easily readable way

Class: MemoryDocument

Model class for a document that is kept in the memory.

Methods:

public MemoryDocument (Byte[] data)

Constructor that creates an object with only a byte array as data

public MemoryDocument (Byte[] data, String name)

Constructor that creates an object with a byte array as data and a name

public MemoryDocument (Byte[] data, String name, MimeType mimeType)

Constructor that creates an object with a byte array as data, a name and a MimeType

public String getName ()

Returns the name of the document

public MimeType getMimeType ()

Returns the MimeType information saved for this object

public InputStream openStream ()

Returns the data as a ByteArrayInputStream

public void save(String filePath)

Saves the MemoryDocument at the given filePath

Enumeration: MimeType

Enumeration that defines the following Mime-Types

```
BINARY      ("application/octet-stream")
XML         ("text/xml")
PDF         ("application/pdf")
PKCS7       ("application/pkcs7-signature")
ASICS       ("application/vnd.etsi.asic-s+zip")
```

Methods:

public String getCode ()

Returns the code for the Mime Type

public static MimeType fromFileName (String filename)

Returns the MimeType for XML, PDF and binary files depending on the file extension

Class: ProxyData

Class that contains the attributes to use a proxy connection

public ProxyData setHost (String host)

Sets the host attribute required for a proxy connection

public String getHost ()

Returns the host attribute

public ProxyData setPort (Int port)

Sets the port attribute required for a proxy connection

public Int getPort ()

Returns the port attribute

public ProxyData setAuthName (String authenticationName)

Sets the authentication name

public String getAuthName ()

Returns the user attribute

public ProxyData setAuthPass (String authenticationPassword)

Sets the password attribute

public String getAuthPass ()

Returns the set authentication password

public boolean hasAuth ()

Checks whether authentication data (name & pass) is provided

public boolean isValid ()

Checks whether the host is not empty and the port is greater than 0

public String toString ()

Overrides the default toString-Method of this object to transport the information of the object in an easily readable way

Class: SignatureParameters

Model class for the attributes required to create a signature.

Methods:

public SignatureParameters ()

Default constructor

public PrivateKey getPrivateKey ()

Returns the private key of the signatory

public SignatureParameters setPrivateKey (PrivateKey privateKey)

Sets the private key of the signatory

public X509Certificate getCertificate ()

Returns the certificate of the signatory

public SignatureParameters setCertificate (X509Certificate certificate)

Sets the signatory's certificate

public X509Certificate[] getCertificateChain ()

Returns the chain of certificates from the signatory up to his root certification authority

public SignatureParameters setCertificateChain (X509Certificate[] certificateChain)

Sets the chain of certificates from the signatory up to his root certification authority

public String getSignatureAlgorithm ()

Returns the signature algorithm used in the signing process

public SignatureParameters setSignatureAlgorithm (String encryptionAlgorithm)

Sets the signature algorithm used in the signing process

public String getDigestAlgorithm ()

Returns the digest algorithm

public SignatureParameters setDigestAlgorithm (String digestAlgorithm)

Sets the digest algorithm

5.2 Package: eu.ecodex.dss.model.checks

This package contains classes that are required for the realisation of a basic test structure.

Interface: Checker <T>

Interface that enables to run tests and check technical rules or business logic

The method in this interface is just a declaration. The actual logic needs to be addressed in the classes that implement this interface.

Methods:

public CheckResult run (T object)

Returns the CheckResult after executing the check

Class: CheckProblem

Model class that represents a problem including the error status (fatal / not fatal) and a message that indicates the problem

Methods:

public CheckProblem (boolean fatal, String message)

Constructor that creates an object with an error status [fatal] and a [message]

public boolean isFatal ()

Returns whether the encountered problem is fatal or not

public String getMessage ()

Returns the message that indicates the problem

Class: CheckResult

Model class that contains the result of a check

Methods:

public CheckResult ()

Default Constructor

public List<CheckProblem> getProblems ()

Returns a copy of the problem list

public CheckResult addProblem (boolean fatal, String message)

Creates a new CheckProblem object with the values [fatal] and [message] and adds it to the problem list

public CheckResult addProblem (CheckProblem problem)

Adds a problem to the problem list

public CheckResult addProblems (List<CheckProblem> problems)

Adds a list of problems to the problem list

public CheckResult addProblems (CheckResult result)

Adds the problems of the check [result] to the problem list

public boolean isSuccessful ()

Checks whether the overall result is successful

public boolean isProblematic ()

Checks whether the overall result is not successful

public boolean isFatal ()

Checks whether the overall result is not successful because of fatal problems

public String toString ()

Overrides the default toString-Method of this object to transport the information of the object in an easily readable way

5.3 Package: eu.ecodex.dss.model.token

This package contains the classes that contribute to the Trust OK-Token

Enumeration: AdvancedSystemType

Enumeration that defines the following types for Advanced Electronic Systems

```
SIGNATURE_BASED       ("Signature-based", "Signature-based")
AUTHENTICATION_BASED  ("Authentication-based", "Authentication-based")
```

Methods:

public AdvancedSystemType (String value, String text)

Constructor that set the provided attributes value and text for the new object

public String getValue ()

Returns the value

public String getText ()

Returns the text

public static AdvancedSystemType fromValue (String value)

Factory retrieval method that returns the matching AdvancedSystemType object

Class: AuthenticationCertificate

Class that contains information about the result from the verification of an authentication certificate against the list of authentication service certificates

Methods:

public AuthenticationCertificate()

Constructor setting the successful validation of the authentication certificate to “false”

public boolean isValidationSuccessful()

Receives information about the validity of an authentication certificate

public void setValidationSuccessful(boolean validationSuccessful)

Sets the validation result for an authentication certificate

Class: AuthenticationInformation

Class that contains information about authentication-based Advanced Electronic Systems

Methods:

public AuthenticationInformation ()

Default constructor

public String getIdentityProvider ()

Returns the name of the identity provider

public AuthenticationInformation setIdentityProvider (String value)

Sets the name of the identity provider to [value]

public String getUsernameSynonym ()

Returns the synonym for the user

public AuthenticationInformation setUsernameSynonym (String value)

Sets the synonym for the user to [value]

public XMLGregorianCalender getTimeOfAuthentication ()

Returns the time of authentication

public AuthenticationInformation setTimeOfAuthentication (XMLGregorianCalendar time)

Sets the time of authentication

Enumeration: LegalTrustLevel

Enumeration that defines the two types of Trust Levels:

```
SUCCESSFUL      ("SUCCESSFUL",      "Successful")
NOTSUCCESSFUL   ("NOT_SUCCESSFUL",  "Not Successful")
```

Methods:

private LegalTrustLevel (String value, String text)

Constructor that sets [value] and [text]

public String getValue ()

Returns the value

public String value ()

Returns the value

public String getText ()

Returns the text

public static LegalTrustLevel fromValue (String value)

Factory retrieval method that returns the matching LegalTrustLevel object

public static boolean isSuccessful (LegalTrustLevel level)

Checks whether [level] is successful

public static boolean isNotSuccessful (LegalTrustLevel level)

Checks whether [level] is not successful

public static LegalTrustLevel worst (LegalTrustLevel... levels)

Returns the worst trust level in the array [levels]

Class: LegalValidationResult

Class that contains information about the legal validation

Methods:

public LegalValidationResult ()

Default constructor

public LegalTrustLevel getTrustLevel ()

Returns the result of the legal evaluation

public LegalValidationResult setTrustLevel (LegalTrustLevel value)

Sets the legal evaluation to [value]

public String getDisclaimer ()

Returns the disclaimer notice

public LegalValidationResult setDisclaimer (String value)

Sets the disclaimer notice to [value]

Class: OriginalValidationReportContainer

Class that contains the details of the original Validation Report

Methods:

public OriginalValidationReportContainer ()

Default constructor

public List<Object> getAny ()

Returns a live list of objects

Subclass: **public static SimpleTypeEntry**

Class that acts as a wrapper for an entry in the live list that allows marshalling and unmarshalling for simple java types

public SimpleTypeEntry ()

Default Constructor

public SimpleTypeEntry (Object value)

Constructor that assigns [value]

public String toString ()

Overrides the default toString-Method of this object to transport the information of the object

Class: Signature

Class that contains information about an applied signature

Methods:

public Signature ()

Default constructor

public AuthenticationCertificate getAuthenticationCertValidation()

Returns the result from the verification of an authentication certificate

public Signature setAuthenticationCertValidation(
AuthenticationCertificate authenticationCertValidation)

Sets the result from the verification of an authentication certificate

public boolean isUnsigned()

Returns the information about whether the document is unsigned

public Signature setUnsigned(boolean value)

Sets the information whether the document is unsigned

public XMLGregorianCalendar getSigningTime ()

Returns the signing time

public Signature setSigningTime (XMLGregorianCalendar value)

Sets the signing time to [value]

public SignatureAttributes getSignatureInformation ()

Returns the signature information

public Signature setSignatureInformation (SignatureAttribute value)

Sets the signature information to [value]

public SignatureCertificate getCertificateInformation ()

Returns the certificate information

public Signature setCertificateInformation (SignatureCertificate value)

Sets the certificate information to [value]

Class: SignatureAttributes

Class that contains detailed information about a signature

Methods:

public SignatureAttributes ()

Default constructor

public SignatureAttributes setSignatureValid(boolean value)

Sets the signature verification property to [value]

public boolean isSignatureValid()

Checks whether the signature is valid

public SignatureAttributes setStructureValid(boolean value)

Sets the structure verification property to [value]

public boolean isStructureValid()

Checks whether the structure of the signature is valid

public String getSignatureFormat ()

Returns the signature format

public SignatureAttributes setSignatureFormat (String value)

Sets the signature format to [value]

public String getSignatureLevel ()

Returns the signature level

public SignatureAttributes setSignatureLevel (String value)

Sets the signature level to [value]

Class: SignatureCertificate

Class that holds information about the certificate used to sign

Methods:

public SignatureCertificate ()

Default constructor

public String getIssuer ()

Returns the issuer of the certificate

public SignatureCertificate setIssuer (String value)

Sets the issuer to [value]

public SignatureCertificate setCertificateValid(boolean value)

Sets the certificate verification property to [value]

public boolean isCertificateValid()

Checks whether the certificate is valid

public SignatureCertificate setValidityAtSigningTime (boolean value)

Sets the certificate validity at the time of signing to [value]

public boolean isValidityAtSigningTime ()

Checks whether the certificate was valid at the time of signing

Enumeration: TechnicalTrustLevel

Enumeration that defines the types of technical Trust Levels:

```
FAIL          ("FAIL", "Failed")
SUFFICIENT    ("SUFFICIENT", "Sufficient")
SUCCESSFUL    ("SUCCESSFUL", "Successful")
```

Methods:

private TechnicalTrustLevel (String value, String text)

Constructor that sets [value] and [text]

public String getValue()

Returns the value

public String value()

Returns the value

public String getText ()

Returns the text

public static TechnicalTrustLevel fromValue (String value)

Factory retrieval method that returns the matching TechnicalTrustLevel object

public static boolean isSuccessful (TechnicalTrustLevel level)

Checks whether [level] is successful

public static boolean isSufficient (TechnicalTrustLevel level)

Checks whether [level] is sufficient

public static boolean isFail (TechnicalTrustLevel level)

Checks whether [level] is neither successful nor sufficient

public static TechnicalTrustLevel worst (TechnicalTrustLevel... levels)

Returns the worst trust level in the array [levels]

Class: TechnicalValidationResult

Class that contains the data about the technical validation

Methods:

public TechnicalValidationResult ()

Default constructor

public TechnicalTrustLevel getTrustLevel ()

Returns the result of the technical evaluation

public TechnicalValidationResult setTrustLevel (TechnicalTrustLevel value)

Sets the technical evaluation to [value]

public String getComment()

Returns the comment of the technical evaluation

public TechnicalValidationResult setComment(String value)

Set the comment attribute of the technical evaluation to [value]

Class: Token

Class that holds the token and acts as the container for all information

Methods:

public Token ()

Default constructor

public TokenIssuer getIssuer ()

Returns information about the token issuer

public Token setIssuer (TokenIssuer value)

Sets the token issuer to [value]

public TokenDocument getDocument ()

Returns the token document

public Token setDocument (TokenDocument value)

Sets the token document to [value]

public TokenValidation getValidation ()

Returns the validation information about the token

public Token setValidation (TokenValidation value)

Sets the token validation information to [value]

Convenience methods to access attributes in the data structure:

public String getIssuerCountry ()

public String getIssuerServiceProvider()

public String getAdvancedElectronicSystem ()

public String getAdvancedElectronicSystemText ()

public String getDocumentName ()

public String getDocumentType ()

public DigestMethodType getDocumentDigestMethod()

public byte[] getDocumentDigestValue()

public TechnicalValidationresult getTechnicalValidationResult ()

public TechnicalTrustLevel getTechnicalValidationResultTrustLevel ()

public String getTechnicalValidationResultComments ()

public LegalValidationResult getLegalValidationResult ()

public LegalTrustLevel getLegalValidationResultTrustLevel ()

public String getLegalValidationResultDisclaimer ()

public OriginalValidationReportContainer getValidationOriginalReport()

public ValidationVerification getValidationVerificationData ()

public Signature getValidationVerificationSignatureData ()

public SignatureCertificate getValidationVerificationSignatureCertificateInformation ()

public String getValidationVerificationSignatureCertificateIssuer ()

public SignatureAttributes getValidationVerificationSignatureInformation ()

public String getValidationVerificationSignatureFormat()

public String getValidationVerificationSignatureLevel ()

public XMLGregorianCalendar getValidationVerificationSignatureSigningTime ()

public XMLGregorianCalendar getValidationVerificationTime ()

public XMLGregorianCalendar getValidationVerificationAuthenticationTime()

public boolean isValidationVerificationSignatureValid()

public boolean isValidationVerificationSignatureUnsigned()

public boolean isValidationVerificationSignatureCertificateValid()

public boolean isValidationVerificationSignatureCertificateValidityAtSigningTime ()

public boolean isValidationVerificationSignatureStructureValid()

public AuthenticationInformation getValidationVerificationAuthenticationData ()

public String getValidationVerificationAuthenticationProvider ()

public String getValidationVerificationAuthenticationUsername ()

Class: TokenDocument

Class that contains the document

Methods:

public TokenDocument ()

Default constructor

public String getFilename ()

Returns the filename

public TokenDocument setFilename (String value)

Sets the filename to [value]

public String getType ()

Returns the type

public TokenDocument setType (String value)

Sets the type property to [value]

public DigestMethodType getDigestMethod ()

Returns the digest method

public TokenDocument setDigestMethod (DigestMethodType digestMethod)

Sets the digest method to [digestMethod]

public byte[] getDigestValue ()

Returns the digest value

public TokenDocument setDigestValue (byte[] digestValue)

Sets the digest value to [digestValue]

public String getSignatureFilename ()

Returns the signature filename

public TokenDocument setSignatureFilename (String value)

Sets the signature filename property to [v]

Class: TokenIssuer

Class that contains information about the token issuer

Methods:

public TokenIssuer ()

Default constructor

public String getServiceProvider ()

Returns the name of the service provider

public TokenIssuer setServiceProvider (String value)

Sets the service provider to [value]

public String getCountry ()

Returns the country of the issuer

public TokenIssuer setCountry (String value)

Sets the country of the issuer to [value]

public AdvancedSystemType getAdvancedElectronicSystem ()

Returns the Advanced Electronic System type

public TokenIssuer setAdvancedElectronicSystem (AdvancedSystemType value)

Set the Advanced Electronic System to [value]

Class: TokenValidation

Class that holds the information about the validation of a token

Methods:

public TokenValidation ()

Default constructor

public XMLGregorianCalendar getVerificationTime ()

Returns the time the token verification was performed

public TokenValidation setVerificationTime (XMLGregorianCalendar value)

Sets the time of the token verification to [value]

public ValidationVerification getVerificationData ()

Returns the verification data

public TokenValidation setVerificationData (ValidationVerification value)

Sets the verification data to [value]

public TechnicalValidationResult getTechnicalResult ()

Returns the technical result

public TokenValidation setTechnicalResult (TechnicalValidationResult value)

Sets the technical result to [value]

public LegalValidationResult getLegalResult ()

Returns the legal result

public TokenValidation setLegalResult (LegalValidationResult value)

Sets the legal result to [value]

public OriginalValidationReportContainer getOriginalValidationReport ()

Returns the validation report container

public TokenValidation setOriginalValidationReport (OriginalValidationReportContainer value)

Sets the validation report container to [value]

Class: ValidationVerification

Class that holds information about verification

Methods:

public ValidationVerification ()

Default constructor

public Signature getSignatureData ()

Returns the signature data

public ValidationVerification setSignatureData (Signature value)

Sets the signature data to [value]

public AuthenticationInformation getAuthenticationData ()

Returns the authentication data

public ValidationVerification setAuthenticationData (AuthenticationInformation value)

Sets the authentication data to [value]

5.4 Package: eu.ecodex.dss.service

Interface: ContainerFileDefinitions

Interface that defines the following constants for locations and filenames

```
SIGNED_CONTENT       FileDef(null,        "SignedContent.zip")
TOKEN_PDF            FileDef(null,        "TrustOkToken.pdf")
TOKEN_XML            FileDef("META-INF",  "trustOkToken.xml")
SIGNATURES           FileDef("META-INF",  "signatures.xml")
SIGNED_CONTENT_ASIC  FileDef("META-INF",  "SignedContent.zip.ASIC")
```

Subclass: **public static FileDef**

Class that allows access to the location and name of a file

public FileDef (String location, String name)

Constructor that sets [location] and [name] and generates the full path

public String getLocation ()

Returns the location of the file inside the ASiC-Container

public String getName ()

Returns the name of the file

public String getReference ()

Returns the full path of the file

Class: ECodexException extends Exception

Class used to indicate library-scoped e-CODEX exceptions

Methods:

public ECodexException ()

Default constructor

public ECodexException (String message)

Constructor that calls the superclass with the attribute [message]

public ECodexException (Throwable cause)

Constructor that calls the superclass with the attribute [cause]

public ECodexException (String message, Throwable cause)

Constructor that calls the superclass with the attributes [message] and [cause]

public static ECodexException wrap (Exception e)

Static method to wrap an Exception [e] into an ECodexException

Class: ECodexBusinessException extends ECodexException

Class used for non-technical exceptions in order to indicate rule violations

Methods:

public ECodexBusinessException (String message, CheckResult checkResult)

Constructor that calls the superclass with [message] and saves [checkResult]

public CheckResult getCheckResult ()

Returns the check result

public String getCheckResultDetails ()

Returns a detailed textual representation for the set check result

public static String createCheckResultDetails (CheckResult checkResult)

Static method to generate a detailed textual representation of [checkResult]

Interface: ECodexContainerService

Interface that declares the required methods for handling the ECodexContainer

The actual logic needs to be addressed in the classes that implement this interface

Methods:

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

Sets [conf] to establish the configuration and update the connector certificates

public void setContainerSignatureParameters (SignatureParameters signingParameters)

Sets [signingParameters] to configure the parameters for signing the ASiC-S container

public void setTechnicalValidationService (ECodexTechnicalValidationService validationService)

Sets the technical validation Service to [validationService]

public void setLegalValidationService (ECodexLegalValidationService validationService)

Sets the legal validation service to [validationService]

public ECodexContainer create (BusinessContent businessContent, TokenIssuer issuer)

Returns the created ASiC-S container for [businessContent] and [issuer]

public ECodexContainer receive (InputStream asicInputStream, InputStream tokenStream)

Returns the ASiC-S container received from [asicInputStream] and [tokenStream]

public CheckResult check (ECodexContainer container)

Returns the result of the integrity check of the ASiC-S container

public ECodexContainer addSignature (ECodexContainer container)

Returns the ASiC-S container after an additional signature is applied to [container]

Interface: ECodexLegalValidationService

Interface that declares the required methods for handling the legal validation

The actual logic needs to be addressed in the classes that implement this interface

Methods:

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

Sets [conf] to establish the configuration and update the connector certificates

public LegalValidationResult create (Token token)

Returns the legal validation result for [token]

Interface: ECodexTechnicalValidationService

Interface that declares the required methods for handling the technical validation

The actual logic needs to be addressed in the classes that implement this interface

Methods:

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

Sets [conf] to establish the configuration and update the connector certificates

public TokenValidation create (Document businessDocument)

Returns the technical validation result for [businessDocument]

public Document createReportPDF (Token token)

Returns the generated technical validation report as PDF to be used as human readable part of the Trust OK-Token

5.5 Package: eu.ecodex.dss.service.checks

This package contains classes that perform checks on objects and required attributes

Class: AbstractChecker<T> implements Checker<T>

Class that provides convenience methods and needs to be used if logging is required

Methods:

protected AbstractChecker ()

Default constructor

protected void detect (CheckResult r, boolean fatal, String message)

Convenience Method to add a problem and to address logging at the same time

Class: BusinessContentChecker implements AbstractChecker<BusinessContent>

Class that checks whether the BusinessContent object meets the requirements

Methods:

public BusinessContentChecker ()

Default constructor

public CheckResult run (BusinessContent object)

Checks [object], its business document, its related signature file (in case of a detached signature) and the set attachments against a set of rules:

- The BusinessContent object must not be null
- The Business document, the optional signature file and the attachment(s) must not be null or empty
- The Business document, the optional signature file and the attachment(s) must have valid filenames that are unique for each BusinessContent context

Returns the result of the performed check

Class: ECodexContainerChecker implements AbstractChecker <ECodexContainer>

Class that checks whether the ECodexContainer object meets the requirements

Methods:

public ECodexContainerChecker ()

Default constructor

public CheckResult run (ECodexContainer object)

Checks [object] and its attributes and returns the result

List of the possible error codes:

CONTAINER_MISSING
CONTAINER_ASIC_MISSING
CONTAINER_ASIC_DATA_MISSING
CONTAINER_BUSINESS_MISSING
CONTAINER_BUSINESS_DATA_MISSING
CONTAINER_SIGNATURE_DATA_MISSING
CONTAINER_TOKENPDF_MISSING
CONTAINER_TOKENPDF_DATA_MISSING
CONTAINER_TOKENXML_MISSING
CONTAINER_TOKENXML_DATA_MISSING
TOKEN_MISSING
TOKEN_DOCUMENT_MISSING
TOKEN_DOCUMENT_FILENAME_MISSING
TOKEN_DOCUMENT_TYPE_MISSING
TOKEN_DOCUMENT_DIGESTMETHOD_MISSING
TOKEN_DOCUMENT_DIGESTVALUE_MISSING
~~*TOKEN_TECHNICAL_VALIDATION_RESULT_MISSING*~~
~~*TOKEN_TECHNICAL_VALIDATION_RESULT_TRUSTLEVEL_MISSING*~~
~~*TOKEN_TECHNICAL_VALIDATION_RESULT_COMMENT_MISSING*~~
~~*TOKEN_LEGAL_VALIDATION_RESULT_MISSING*~~
~~*TOKEN_LEGAL_VALIDATION_RESULT_TRUSTLEVEL_MISSING*~~
~~*TOKEN_LEGAL_VALIDATION_RESULT_DISCLAIMER_MISSING*~~
TOKEN_ISSUER_MISSING
TOKEN_ISSUER_SYSTEMTYPE_MISSING
TOKEN_VALIDATION_MISSING
TOKEN_VALIDATION_TIME_MISSING
TOKEN_VALIDATION_TECHNICAL_RESULT_MISSING
TOKEN_VALIDATION_TECHNICAL_RESULT_TRUSTLEVEL_MISSING
TOKEN_VALIDATION_TECHNICAL_RESULT_COMMENT_MISSING
TOKEN_VALIDATION_LEGAL_RESULT_MISSING
TOKEN_VALIDATION_LEGAL_RESULT_TRUSTLEVEL_MISSING
TOKEN_VALIDATION_LEGAL_RESULT_DISCLAIMER_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_AUTHINFO_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_AUTHINFO_USERNAME_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_AUTHINFO_IDENTITYPROVIDER_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_AUTHINFO_TIME_MISSING

TOKEN_VALIDATION_VERIFICATIONDATA_SIGDATA_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_SIGDATA_CERTINFO_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_SIGDATA_CERTINFO_ISSUER_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_SIGDATA_SIGINFO_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_SIGDATA_SIGINFO_FORMAT_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_SIGDATA_SIGINFO_LEVEL_MISSING
TOKEN_VALIDATION_VERIFICATIONDATA_SIGDATA_TIME_MISSING

Class: TokenIssuerChecker implements AbstractChecker <TokenIssuer>

Class that checks whether the TokenIssuer object meets the requirements

Methods:

public TokenIssuerChecker ()

Default constructor

public CheckResult run (TokenIssuer object)

Checks [object] and its attributes against a set of rules:

- The TokenIssuer object must not be null
- Service provider, country and advanced electronic system must not be null or empty
- The Country must be a valid 2-letter country code defined in ISO 3166

Returns the result of the performed check

5.6 Package: eu.ecodex.dss.service.impl

This package contains classes that implement the functionality

Class: ConnectorCertificateStore

Class that accesses a keystore, extracts its certificates into a cache and checks whether a certificate is a connector certificate. The initialised keystore is not automatically updated

Methods:

public ConnectorCertificateStore ()

Default constructor

public synchronized int update (~~EnvironmentConfiguration.~~CertificateStoreInfo info)

Returns the numbers of X509 certificates that are extracted and cached from [info], returns -1 if no keystore information is set or the keystore information is invalid

public boolean isValid (X509Certificate cert)

Checks whether [cert] is a connector certificate by accessing the keystore

Class: DocumentWrapperDSS2ECodex implements eu.ecodex.dss.model.Document

Wrapper class that encapsulates the functionality of the document class to establish a bridge between the core DSS model and the e-CODEX model

Methods:

public DocumentWrapperDSS2ECodex (
eu.europa.ec.markt.dss.signature.DSSDocument document)

Constructor that wraps [d] in eu.ecodex.dss.model.Document

public InputStream openStream ()

Returns the input stream of the document set in the constructor

public String getName ()

Returns the name of the document set in the constructor

public MimeType getMimeType ()

Returns the mime type of the document set in the constructor

Class: DocumentWrapperECodex2DSS

implements eu.europa.ec.markt.dss.signature.DSSDocument

Wrapper class that encapsulates the functionality of the document class to establish a bridge between the e-CODEX model and the core DSS model.

Methods:

public DocumentWrapperECodex2DSS (eu.ecodex.dss.model.Document document)

Constructor that wraps [d] in eu.europa.ec.markt.dss.signature.Document

public InputStream openStream () throws DSSException

Returns the input stream of the document set in the constructor

public byte[] getBytes() throws DSSException

Returns the byte array of the document set in the constructor

public String getName ()

Returns the name of the document set in the constructor

public String getAbsolutePath()

Overwritten method. Returns the name of the document set in the constructor

public MimeType getMimeType ()

Returns the mime type of the document set in the constructor

Class: DSSECodexContainerService implements ECodexContainerService

Class that provides the DSS implementation of the e-CODEX container service

Methods:

public DSSECodexContainerService ()

Default constructor

public void setProcessExecutor(ProcessExecutor processExecutor)

Sets a process executor for the DSS library

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

Sets the environment configuration to [conf]

public void setContainerSignatureParameters (SignatureParameters signingParameters)

Sets the signature parameters to [signingParameters]

public void setCertificateVerifier (CertificateVerifier certificateVerifier)

Sets the certificate verifier to [certificateVerifier]

public void setTechnicalValidationService (ECodexTechnicalValidationService validationService)

Sets the technical validation service to [validationService]

public void setLegalValidationService (ECodexLegalValidationService validationService)

Sets the legal validation service to [validation service]

public ECodexContainer addSignature (ECodexContainer container) throws ECodexException

Attaches an additional XAdES signature based on the set / active signature parameters for the signed content [container] to the signatures.xml

public CheckResult check (ECodexContainer container) throws ECodexException

Checks [container] for strict compliance of the container and the content and aborts further processing in case of a problem.

This consists of integrity checks of container, business content and issuer and validation of the signatures on Trust OK-Token XML, Trust OK-Token PDF and the ASiC document

public ECodexContainer create (BusinessContent businessContent, TokenIssuer issuer)
throws ECodexException

Creates and returns the ECodexContainer from [businessContent] and [issuer]

Overview about the steps in this method:

- Checks whether the legal and technical validation service are set
- Checks whether the legal and technical PDF generators are set
- Checks whether the business content and the token issuer are set and valid
- Creates the token with the provided data
- Creates the technical PDF report and the legal summary
- Creates and signs the Trust OK-Token PDF and XML
- Creates and signs the ASiC document / container
- Returns the container with all documents

public ECodexContainer receive (InputStream asicInputStream, InputStream tokenStream)
throws ECodexException

Generates an ECodexContainer from [asicInputStream] and [tokenStream]

Overview about the steps in this method:

- Checks whether [asicInputStream] is set and a zip document
- Checks whether [tokenStream] is set and in XML format
- Decodes [tokenStream] to get a token
- Strips down [asicInputStream] to document level
- Returns the container with all documents

Class: DSSECodexLegalValidationService implements ECodexLegalValidationService

Class that provides the DSS implementation of the e-CODEX legal validation service

Methods:

public DSSECodexLegalValidationService ()

Default constructor

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

Currently unused method

public LegalValidationResult create (Token token) throws ECodexException

Evaluates [token] and checks it against a set of rules:

- The Token must not be null
- The Token must have a Token Validation object
- The Token Validation must have verification data

The LegalValidationResult is created based on the technical result:

Technical Trust Level	**Legal Trust Level**
Successful	*Successful*
Sufficient	*Not successful*
Fail	*Not successful*

Table 3: Mapping of Technical Trust Level to Legal Trust Level

In addition, in case of an authentication-based system with signature, the result of the certificate verification is taken into account:

Certificate Verification Result	**Legal Trust Level**
Successful	*Result taken from Table 3*
Fail	*Not successful*

Table 4: Mapping of Certificate Verification Result to Legal Trust Level

Class: DSSECodexTechnicalValidationService implements ECodexTechnicalValidationService

Class that provides the DSS implementation of the e-CODEX technical validation service

Methods:

public DSSECodexTechnicalValidationService ()

Default constructor

public void setCertificateVerifier (CertificateVerifier certificateVerifier)

Sets the certificate verifier to [certificateVerifier]

public void setProxyPreferenceManager (ProxyPreferenceManager pPM)

Sets the proxy preference manager to [pPM]

public void setEnvironmentConfiguration (EnvironmentConfiguration conf)

Sets the proxy preference manager configuration to the proxy configuration of [conf]

public TokenValidation create (Document businessDocument, Document detachedSignature)

Creates a Token Validation using the run-Method of the DSSTokenValidationCreator with the set certificate verifier, [businessDocument] and [detachedSignature] and returns it

public Document createReportPDF (Token token)

Creates a PDF report for the original validation report of [token]

During the creation process, [token] is checked against a set of rules:

- The Token must not be null
- The Token must have a Token Validation object
- The Token Validation must have verification data
- The Token Validation must have exactly one validation report

Returns a MemoryDocument ("dss-report.pdf") that contains the Report

public void setProcessExecutor(ProcessExecutor processExecutor)

Sets a process executor for the DSS library

public void initAuthenticationCertificateVerification() throws ECodexException

Initializes the TSL of authentication service certificates

public void isAuthenticationCertificateLOTL(boolean isLOTL)

Marks the TSL of authentication service certificates as “list of the lists”, a list containing multiple lists of authentication service certificates.

protected AuthenticationCertificate verifyAuthenticationCertificate(
final Document businessDocument,
final Document detachedSignature) throws ECodexException

Verifies the certificate of a given signature against an initialized TSL of authentication service certificates

public void setAuthenticationCertificateTSL(String authenticationCertificateTSL)
public void setAuthenticationCertificateTSL(InputStream authenticationCertificateTSL)
public void setAuthenticationCertificateTSL(byte[] authenticationCertificateTSL)

Sets the TSL of authentication service certificates

Class: DSSTokenValidationCreator

Thread-safe class that creates the token validation object

Methods:

~~public~~ DSSTokenValidationCreator (CertificateVerifier cv, Document bd,
Document ds, ProcessExecutor pe)
Constructor that sets the certificate verifier [cv], the business document [bd] and the detached signature [ds]

public TokenValidation getResult ()

Returns the result of the token validation

~~public~~ void run () throws Exception

Method to initiate the creation of the token validation if it has not been created already

Overview about the steps in this method:

- Creates the validation report
- Validates the document and generates the validation report
- Adds the original report to the token validation
- Sets the verification time
- Detects the significant signatures (currently: the last signature applied)
- Sets the signing time
- Sets the certificate verification of the signing certificate
- Sets the issuer of the signing certificate
- Sets the validity at signing time of the signing certificate
- Sets the signature format
- Sets the signature level
- Sets the signature verification
- Sets the signature structure verification
- Determines the result

public static DecisionData getCachedDecisionData ()

Gives access to the latest data used for computing the decision in the current thread

Subclass: **public static class DecisionData**

Class that provides the data that is used to take decisions

~~public~~ DecisionData (DiagnosisData diagnosis, ValidationData validation)

Constructor that sets [diagnosis] and [validation]

public DiagnosisData getDiagnosis ()

Returns the diagnosis data

public ValidationData getValidation ()

Returns the validation data

public TechnicalTrustLevel getLevel ()

Returns the technical trust level

public String toString

Overrides the default toString-Method of this object to transport the information of the object in an easily readable way

Subclass: **public static class DiagnosisData**

Class that provides the data that is used for the validation

~~public~~ Diagnosis Data (XMLGregorianCalendar signingTime, X509Certificate signingCertificate, String signingCertificateIssuer, String signatureFormatLevel, SignatureType signatureConclusion, X509Certificate issuerCertificate)

Constructor that sets the attributes

public String toString

Overrides the default toString-Method of this object to transport the information of the object in an easily readable way

Subclass: **public static class ValidationData**

Class that provides the results of the validation

~~public~~ ValidationData (boolean signatureComputation, boolean signatureConclusion, boolean signatureFormat, TechnicalTrustLevel signatureCertStatus, TechnicalTrustLevel signatureCertHistory, boolean trustAnchor, TechnicalTrustLevel issuerCertStatus, TechnicalTrustLevel issuerCertHistory)

Constructor that sets the different attributes

public String toString ()

Overrides the default toString-Method of this object to transport the information of the object in an easily readable way

Subclass: **public static class** ~~SignatureInformationComparator~~SignatureTimeComparator

Utility class that contains the methods to compare the signing time of SignatureInformation objects by implementing Comparator<SignatureInformation> and create a sorted list

public int compare (SignatureInformation o1, SignatureInformation o2)

Returns the result of the comparison of the signing time of [o1] and [o2]:

Value < 0 if the signing time of [o1] is earlier than the signing time of [o2]

Value 0 if both signing times are identical

Value > 0 if the signing time of [o1] is after the signing time of [o2]

public static List<~~SignatureInformation~~AdvancedSignature> createSortedList (List< ~~SignatureInformation~~AdvancedSignature > ~~infos~~signatures)

Static method that creates a sorted copy of the list [~~infos~~signatures] and returns it

public static void sort (List<~~SignatureInformation~~AdvancedSignature > ~~infos~~signatures)

Static method that sorts the list [~~infos~~signatures]

public static AdvancedSignature getFirst(final List<AdvancedSignature> infos)

Returns the first entry of the list [infos]

public static AdvancedSignature getLast(final List<AdvancedSignature> signatures)

Returns the first entry of the list [signatures]

Class: SigningUtil

Utility class that provides different methods to sign documents

Methods:

~~public~~ static Document signASiC (SignatureParameters signingParams, Document document) throws Exception

Static method to sign the [document] with an ASiC-S BES / detached signature with the signature parameters set in [signingParams]

~~public~~ static Document signPADES (SignatureParameters signingParams, Document document) throws Exception

Static method to sign the [document]with a PAdES BES / enveloped signature with the signature parameters set in [signingParams]

~~public~~ static Document signXADES (SignatureParameters signingParams, Document document, SignaturePackaging signaturePackaging) throws Exception

Static method to sign the [document] with a XAdES BES signature with the signature parameters set in [signingParams] and the packaging options from [signaturePackaging]

Class: TechnicalValidationUtil

Class that provides convenience methods for the validation report

Methods:

public TechnicalValidationUtil ()

Default Constructor

public static CertificateToken getSigningCertificateToken(final AdvancedSignature signature)

Returns the SigningCertificateToken of the [signature]

public static X509Certificate getSigningCertificate (~~SignatureLevelAnalysis sigLevelAnalysis~~ final CertificateToken certificateToken)

Returns the certificate used for signing that part of [sigLevelAnalysis]

public static String getSigningCertificateIssuerName (~~SignatureLevelAnalysis sigLevelAnalysis~~ final X509Certificate certificate)

Returns the issuer name from the signing certificate

public static CertificateToken getIssuerCertificateToken(final CertificateToken certificateToken)

Returns the issuerCertificateToken of the [certificateToken]

public static X509Certificate getIssuerCertificate (~~CertPathRevocationAnalysis certPathRevoAnalysis, X509Certificate signingCertificate~~ final CertificateToken certificateToken)

Returns the issuer certificate of [signingCertificate] from [certPathRevoAnalysis]

public static XMLGregorianCalendar getSigningTime (~~SignatureLevelAnalysis sigLevelAnalysis~~ final DiagnosticData diagnosticData, final String signatureId)

Returns the signing time saved in [sigLevelAnalysis]

~~public static String getSignatureFormat (SignatureLevelAnalysis signatureLevelAnalysis)~~

~~Returns the signature format from [signatureLevelAnalysis]~~

~~public static String getSignatureLevel (SignatureInformation sigInfo)~~

~~Returns the signature level from [sigInfo]~~

public static String getSignatureFormatLevel (~~SignatureLevelAnalysis signatureLevelAnalysis~~ final SimpleReport simpleReport, final String signatureId)

Returns the signature format and the level from [signatureLevelAnalysis]

public static ~~FinalConclusion~~ SignatureType getSignatureConclusion (~~SignatureInformation sigInformation~~ final SimpleReport simpleReport, final String signatureId)

Returns the final conclusion saved in the [sigInformation]

public static boolean checkSignatureCorrectness (~~SignatureVerification signatureVerification~~ final SimpleReport simpleReport, final String signatureId)

Checks whether the signature is mathematically correct

public static TechnicalTrustLevel checkCertificateRevocation (~~CertPathRevocationAnalysis certPathRevocationAnalysis, X509Certificate signingCertificate~~ final CertificateToken certificateToken)

Returns the technical trust level after searching [signingCertificate] in [certPathRevocationAnalysis] and comparing them

public static TechnicalTrustLevel checkCertificateValidity (~~CertPathRevocationAnalysis certPathRevonAnalysis, X509Certificate certificate~~final CertificateToken certificateToken, XMLGregorianCalendar signingTime)

Returns the technical trust level after searching the signing certificate [certificate] in [certPathRevocationAnalysis] and checking the validity of [certificate] at [signingTime]

~~public static TechnicalTrustLevel checkCertificateValidityAtTime (X509Certificate certificate, XMLGregorianCalendar time)~~

~~Returns the technical trust level after checking the validity of [certificate] at [time]~~

public static boolean checkTrustAnchor (~~CertPathRevocationAnalysis certPathRevoAnalysis~~ final CertificateToken certificateToken)

Checks whether [certPathRevoAnalysis] contains trusted list information and the service can be found

~~Class: TokenStreamUtil~~

~~Utility class to encode and decode a token using a static JAXBContext to ensure thread safety~~

~~Methods:~~

~~public static Token decodeXMLStream (InputStream xmlInputStream)~~

~~Static method that decodes [xmlInputStream] to a token~~

~~public static ByteArrayOutputStream encodeXMLStream (Token token)~~

~~Static method that encodes [token] to a XML stream~~

5.7 Package: eu.ecodex.dss.util

This package contains utility classes to provide general useful functions.

Abstract Class: AbstractPDFGenerator

Abstract class to provide basic resources and functionality for the generation of pdf documents

Methods:

public abstract Document generate (Token token)

Declaration of the method to generate the pdf document for [token]

Class: DigestUtil

Class that provides convenience methods to generate digests

Methods:

public static byte[] digest (byte[] bytes, DigestAlgorithm algorithm)

Static method that returns a base64 encoded hash value for [bytes] using [algorithm]

public static byte[] digest (byte[] bytes, String algorithm)

Static method that returns a base64 encoded hash value for [bytes] using [algorithm]

public static byte[] digestSHA256 (byte[] bytes)

Static method that returns a base64 encoded hash value for [bytes] using SHA256

Class: DocumentStreamUtil

Class that provides convenience methods for documents

Methods:

public static boolean hasData (Document document)

Checks whether the document [document] exists and is not empty

public static byte[] getData (Document document)

Returns the complete content of the document [document] as byte array

Class: LogDelegate

Class that encapsulates the logging and provides convenience methods

Methods:

public LogDelegate (Class<?> clazz)

Constructor that initialises a logger for the class [clazz]

protected String prepareMessage (String message, boolean detectMethod)

Method that pre-concatenates the class name to [message]

public void mEnter (String method, Object... parameters)

Method used to signal the entering of a method

public void mExit (String method, Object... parameters)

Method used to signal the successful exiting of a method

public void mCause (String method, Throwable cause, Object... parameters)

Method used to signal a problem during the execution of a method

public void lConfig (String message, Object... parameters)

Method used to log the configuration

public void lError (String message, Throwable cause, Object... parameters)

Method used to log errors

public void lError (String message, Object... parameters)

Method used to log errors

public void lWarn (String message, Object... parameters)

Method used to log warnings

public void lInfo (String message, Object... parameters)

Method used to log information

public void lDetail (String message, Object... parameters)

Method used to log detailed information

Class: MemoryProxyDao implements ProxyDao

Class that generates a MemoryProxyDao object

Methods:

public MemoryProxyDao()

Default constructor

public Collection<ProxyPreference> getAll()

Returns the current configuration parameters

public void update(final ProxyPreference proxyPreference)

Update the information of the MemoryProxyDao with the content of [proxyPreferences]

public String toString()

Overrides the default toString-Method of this object to transport the information of the object in an easily readable way

Class: PDFGeneratorLegalSummary

Class that generates the legal summary page of the Trust OK-Token

Methods:

public PDFGeneratorLegalSummary ()

Default constructor

public Document generate (Token token) throws DocumentException

Generates and returns the legal summary page token-summary-legal.pdf

Class: PDFGeneratorTechnicalSummary

Class that generates the technical summary page of the Trust OK-Token

Methods:

public PDFGeneratorTechnicalSummary ()

Default constructor

public Document generate (Token token) throws DocumentException

Generates and returns the technical summary page token-summary-technical.pdf

Class: PDFUtil

Class that defines absolute values and provides convenience methods for the generation of the PDF documents

Attributes:

```
public static String DATE_PATTERN   = “yyyy-MM-dd hh:mm”
public static String REF_FONTS      = “/eu/ecodex/dss/fonts/”
public static String REF_IMAGES     = “/eu/ecodex/dss/images/”
```

public enum Font

Enumeration that defines different fonts to be used

```
LIBERATION_REGULAR          ("LiberationSans-Regular.ttf")
LIBERATION_BOLD_ITALIC      ("LiberationSans-BoldItalic.ttf")
LIBERATION_BOLD             ("LiberationSans-Bold.ttf")
LIBERATION_ITALIC           ("LiberationSans-Italic.ttf")
```

public enum Image

Enumeration that defines filenames for images and logos to be used

```
LOGO_ECODEX                 ("pdf_logo_ecodex.jpg")
LOGO_CIP                    ("pdf_logo_cip.png")

TECHNICAL_FAIL              ("pdf_icon_technical_fail.png")
TECHNICAL_SUFFICIENT        ("pdf_icon_technical_sufficient.png")
TECHNICAL_SUCCESSFUL       ("pdf_icon_technical_successful.png")

LEGAL_NOTSUCCESSFUL         ("pdf_icon_legal_notsuccessful.png")
LEGAL_SUCCESSFUL            ("pdf_icon_legal_successful.png")
```

Methods:

public static com.lowagie.text.Font createFont (Font font, int size)

Creates a com.lowagie.text.Font using `REF_FONTS`, the font [font] and the [size]

public static com.lowagie.text.Font createFont (String name, int size)

Creates a com.lowagie.text.Font using `REF_FONTS`, the font name [name] and the [size]

public static com.lowagie.text.Image createImage (Image image)

Creates a com.lowagie.text.Image using `REF_IMAGES` and the file [image]

public static com.lowagie.text.Image createImage (String name)

Creates a com.lowagie.text.Image using `REF_IMAGES` and the filename [name]

public static String format (XMLGregorianCalendar cal)

Formats the date [cal] to String

public static String format (String text)

Formats the string [text] and ensures that it cannot be exposed as null

public static int ~~countPages~~ getPageCount (Document document)

Returns the number of pages the PDF document [document]

public static Document concatenate (String filename, Document ... documents)

Concatenates all files in [documents] in given order into a single file [filename]

public static boolean isPDFFile (Document document)

Checks whether [document] is a valid PDF file or not

public static boolean isPDFContent(final Document document)

Overwritten method: Checks whether [document] is a valid PDF file and has at least one page of content

Class: PdfValidationReportService

Class that creates a PDF report from a validation report of a document

Methods:

public PdfValidationReportService()

Default constructor

public void createReport(DiagnosticData diagnosticData,
SimpleReport simpleReport, OutputStream pdfStream)

Create the report with [diagnosticData] and [simpleReport] as source of information

Class: ResourceUtil

Utility class to store a document

Methods:

public PdfValidationReportService()

Default constructor

public static URL getURL(final String name)

Returns a URL object for the resource [name]

public static byte[] getBytes(final String name)

Create an InputStream to [name] and return the respective content as byte array

public static InputStream getStream(final String name)

Return an InputStream to [name]

Class: SignatureParametersFactory

Class that provides convenience methods for creating a SignatureParameters instance

Methods:

public static SignatureParameters create (~~EnvironmentConfiguration.~~CertificateStoreInfo certStoreInfo, String certPassword)

Static method to conveniently create a SignatureParameters object by using external algorithm settings

public static SignatureParameters create (~~EnvironmentConfiguration.~~CertificateStoreInfo certStoreInfo, String certAlias, String certPassword, ~~SignatureAlgorithm algoSignature~~ EncryptionAlgorithm encryptionAlgorithm, DigestAlgorithm algoDigest)

Static method to conveniently create a SignatureParameters object

Class: TokenJAXBObjectFactory

Class that contains factory methods for each Java content interface and Java element interface generated in the eu.ecodex.dss.model.token package

Methods:

public TokenJAXBObjectFactory ()

Default constructor

Factory methods:

public Token createToken ()

public SignatureAttributes createSignatureAttributes ()

public TokenIssuer createTokenIssuer ()

public ValidationVerification createValidationVerification ()

public AuthenticationInformation createAuthenticationInformation ()

public TokenValidation create TokenValidation ()

public OriginalValidationReportContainer createOriginalValidationReportContainer ()

public SignatureCertificate createSignatureCertificate ()

public Signature createSignature ()

public TokenDocument createTokenDocument ()

public TechnicalValidationResult createTechnicalValidationResult ()

public LegalValidationResult createLegalValidationResult ()

public JAXBElement<Token> createTrustOkToken (Token value)

Class: TokenStreamUtil

Utility class to encode and decode a token using a static JAXBContext to ensure thread safety

Methods:

public static Token decodeXMLStream (InputStream xmlInputStream) throws Exception

Static method that decodes [xmlInputStream] to a token

public static ByteArrayOutputStream encodeXMLStream (Token token) throws Exception

Static method that encodes [token] to a XML stream

Class: TokenXMLValidatorUtil

Utility class to validate a token xml file

Methods:

public static boolean isTokenSchemaValid(final Document document)

Validates a token xml file

Class: XmlStreamUtil

Class that provides convenience methods for handling XML streams

Methods:

public static boolean isXmlFile (Document document)

Static method to check whether [document] is a well-formed XML file

Class: ZipStreamUtil

Class that provides convenience methods for handling ZIP documents

Methods:

public static boolean isZipFile (Document zipDocument)

Static method to check whether the document [zipDocument] is a ZIP file or not

public static List<Document> extract (Document zipDocument) throws IOException

Static method to extract all documents contained in the document [zipDocument]

public static Document extract (Document zipDocument, String name) throws IOException

Static method to extract the specific document [name] from the archive [zipDocument]

5.8 Package: eu.ecodex.dss.util.tsl

This package contains utility classes to provide useful functions for TSL handling.

Class: ReactiveDataLoader implements DataLoader

Class to provide a DataLoader that reacts on various sources

Methods:

public ReactiveDataLoader(Document inMemoryTSL, Object authenticationCertificateTSL, ProxyPreferenceManager proxyManager)

Default constructor

public byte[] get(String givenURL)

Open a stream to [givenURL] and return the received content as byte array

Currently, the following URI schemes are supported:

- https:
- file:
- inmemory:bytearray (Mostly for internal use)
- inmemory:inputstream (Mostly for internal use)

public byte[] post(String URL, byte[] content)

Open a stream to [givenURL] and return the received content as byte array

Currently, the following URI schemes are supported:

- https: