Digital Signature Service

Table of Contents

Introduction . . . 3
- Purpose of the document . . . 3
- Scope of the document . . . 3
- Abbreviations and Acronyms . . . 4
- References . . . 6
- Useful links . . . 7

General framework structure . . . 8
- Maven modules . . . 8
- DSS Utils . . . 13
- DSS CRL Parser . . . 13
- DSS PAdES . . . 14

Available demonstrations . . . 15
Signature’s profile simplification . . . 15
- Signature profile guide . . . 17

The XML Signature (XAdES) . . . 19
- XAdES Profiles . . . 19
- Various settings . . . 36
- Multiple signatures . . . 39
- The XML Signature Extension (XAdES) . . . 39
- XAdES-BASELINE-T . . . 39
- XAdES-BASELINE-LT and -LTA . . . 40
- XAdES and specific schema version . . . 41

The signature validation . . . 41
- Validation Process . . . 41
- Validation Result Materials . . . 46
- Customised Validation Policy . . . 82

CAdES signature (CMS) . . . 89
PAdES signature (PDF) . . . 90
- PAdES Visible Signature . . . 92

ASiC signature (containers) . . . 99
Timestamps . . . 102
- Timestamp creation . . . 102
- Timestamp validation . . . 102

Available implementations of DSSDocument . . . 108
Management of signature tokens . . . 108
- PKCS#11 . . . 108

PKCS#12 . . . 109
MS CAPI . . . 109
Other Implementations . . . 110
Management of certificates sources . . . 110
Management of CRL and OCSP sources . . . 111
Repository Revocation Source . . . 112
Other implementations of CRL and OCSP Sources . . . 114
CertificateVerifier configuration . . . 116
Trust Anchor(s) configuration . . . 118
Trust store initialization . . . 118
Trusted List Certificate Source . . . 119
TLValidationJob . . . 120
TLSource and LOTLSource . . . 120
DSSFileLoader . . . 122
The SynchronizationStrategy . . . 123
The CacheCleaner . . . 124
Alerting from TL Loading . . . 124
Executor Service . . . 125
Complete configuration for the European LOTL . . . 126
The TL / LOTL refresh . . . 128
TLValidationJobSummary . . . 130
TSP Sources . . . 130
Time-stamp policy . . . 131
Composite TSP Source . . . 131
Supported algorithms . . . 132
Multi-threading . . . 133
Resource sharing . . . 133
Caching . . . 134
JAXB modules . . . 134
Report stylesheets . . . 136
I18N (Internationalization) . . . 137
Additional features . . . 137
Certificate validation . . . 137
Extract the signed data from a signature . . . 138
REST and SOAP Services . . . 139
REST signature service . . . 140
REST server signature service . . . 157
REST validation service . . . 161
REST certificate validation service . . . 188
REST remote timestamp service . . . 194

Introduction

Purpose of the document

This document describes some examples of how to develop in Java using the DSS framework. The aim is to show to the developers, in a progressive manner, the different uses of the framework. It will familiarize them with the code step by step.

Scope of the document

This document provides examples of code which allow easy handling of digital signatures. The examples are consistent with the Release 5.6 of DSS framework which can be downloaded via https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/DSS+releases

Three main features can be distinguished within the framework :

- The digital signature;
- The extension of a digital signature and;
- The validation of a digital signature.

On a more detailed manner the following concepts and features are addressed in this document:

- Formats of the signed documents: XML, PDF, DOC, TXT, ZIP…;
- Packaging structures: enveloping, enveloped, detached and internally-detached;
- Forms of digital signatures: XAdES, CAdES, PAdES and ASiC-S/ASiC-E;
- Profiles associated to each form of the digital signature;
- Trust management;
- Revocation data handling (OCSP and CRL sources);
- Certificate chain building;
- Signature validation and validation policy;
- Signature qualification;
- Validation reports (Simple, Detailed, ETSI Validation report);
- Management of signature tokens;
- Validation of the signing certificate;
- Timestamp creation;
- Timestamp validation and qualification;
- REST and SOAP webservices.

This is not an exhaustive list of all the possibilities offered by the framework and the proposed examples cover only the most useful features. However, to discover every detail of the operational principles of the framework, the JavaDoc is available within the source code.

Please note that the DSS framework is still under maintenance and new features will be released in the future.

Abbreviations and Acronyms

Table 1. Abbreviations and Acronyms

Code	Description
AdES	Advanced Electronic Signature
API	Application Programming Interface
ASiC	Associated Signature Containers
BB	Building Block (CEF)
CA	Certificate authority
CAdES	CMS Advanced Electronic Signatures
CD	Commission Decision
CEF	Connecting Europe Facility
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
CSP	Core Service Platform (CEF)
CSP	Cryptographic Service Provider
DER	Distinguished Encoding Rules
DSA	Digital Signature Algorithm - an algorithm for public-key cryptography
DSI	Digital Service Infrastructure (CEF)
DSS	Digital Signature Service
EC	European Commission
eID	Electronic Identity Card
ESI	Electronic Signatures and Infrastructures
ETSI	European Telecommunications Standards Institute
EUPL	European Union Public License
FSF	Free Software Foundation
GS	Generic Service (CEF)
GUI	Graphical User Interface
HSM	Hardware Security Modules
HTTP	Hypertext Transfer Protocol
I18N	Internationalization

Java EE	Java Enterprise Edition
JavaDoc	JavaDoc is developed by Sun Microsystems to create API documentation in HTML format from the comments in the source code. JavaDoc is an industrial standard for documenting Java classes.
JAXB	Java Architecture for XML Binding
JCA	Java Cryptographic Architecture
JCE	Java Cryptography Extension
JDBC	Java DataBase Connectivity
LGPL	Lesser General Public License
LOTL	List of Trusted List or List of the Lists
LSP	Large Scale Pilot
MIT	Massachusetts Institute of Technology
MOCCA	Austrian Modular Open Citizen Card Architecture; implemented in Java
MS / EUMS	Member State
MS CAPI	Microsoft Cryptographic Application Programming Interface
OCF	OEBPS Container Format
OCSP	Online Certificate Status Protocol
ODF	Open Document Format
ODT	Open Document Text
OEBPS	Open eBook Publication Structure
OID	Object Identifier
OOXML	Office Open XML
OSI	Open Source Initiative
OSS	Open Source Software
PAdES	PDF Advanced Electronic Signatures
PC/SC	Personal computer/Smart Card
PDF	Portable Document Format
PDFBox	Apache PDFBox - A Java PDF Library: http://pdfbox.apache.org/
PKCS	Public Key Cryptographic Standards
PKCS#12	It defines a file format commonly used to store X.509 private key accompanying public key certificates, protected by symmetrical password

PKIX	Internet X.509 Public Key Infrastructure
RSA	Rivest Shamir Adleman - an algorithm for public-key cryptography
SCA	Signature Creation Application
SCD	Signature Creation Device
SME	Subject Matter Expert
SMO	Stakeholder Management Office (CEF)
SOAP	Simple Object Access Protocol
SSCD	Secure Signature-Creation Device
SVA	Signature Validation Application
TL	Trusted List
TLManager	Application for managing trusted lists.
TSA	Time Stamping Authority
TSL	Trust-service Status List
TSP	Time Stamp Protocol
TSP	Trusted Service Provider
TST	Time-Stamp Token
UCF	Universal Container Format
URI	Uniform Resource Identifier
WSDL	Web Services Description Language
WYSIWYS	What you see is what you sign
XAdES	XML Advanced Electronic Signatures
XML	Extensible Markup Language
ZIP	File format used for data compression and archiving

References

Table 2. References

Ref.	Title	Reference	Version
R01	ESI - XAdES digital signatures	ETSI EN 319 132 part 1-2	1.1.1
R02	ESI - CAdES digital signatures	ETSI EN 319 122 part 1-2	1.1.1
R03	ESI - PAdES digital signatures	ETSI EN 319 142 part 1-2	1.1.1

R04	ESI - Associated Signature Containers (ASiC)	ETSI EN 319 162 part 1-2	1.1.1
R05	Document management - Portable document format - Part 1: PDF 1.7	ISO 32000-1	1
R06	Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.	DIRECTIVE 1999/93/EC	
R07	Internet X.509 Public Key Infrastructure - Time-Stamp Protocol (TSP)	RFC 3161	
R08	ESI - Procedures for Creation and Validation of AdES Digital Signatures	ETSI EN 319 102-1	1.1.1
R09	ESI - Signature validation policy for European qualified electronic signatures/seals using trusted lists	ETSI TS 119 172-4	draft
R10	ESI - Trusted Lists	ETSI TS 119 612	2.1.1
R11	eIDAS Regulation No 910/2014	910/2014/EU	
R12	ESI - Procedures for Creation and Validation of AdES Digital Signatures	ETSI TS 119 102-2	1.2.1
R13	ESI - Procedures for using and interpreting EU Member States national trusted lists	ETSI TS 119 615	draft

Useful links

- CEF Digital

- eSignature FAQ
- TL Browser
- eSignature validation tests
- Trusted List Manager non-EU
- Source code (GitHub)
- Source code (EC Bitbucket)
- Source code demonstrations (EC Bitbucket)
- Report an issue (EC Jira)
- Old Jira

General framework structure

DSS framework is a multi-modules project which can be builded with Maven.

You can easily download them with the following Maven repository :

```
<repository>
  <id>cefdigital</id>
  <name>cefdigital</name>
  <url>
https://ec.europa.eu/cefdigital/artifact/content/repositories/esignaturedss/</url>
</repository>
```

Maven modules

Shared modules

dss-enumerations

Contains a list of all used enumerations in the DSS project.

dss-jaxb-parsers

Contains a list of all classes used to transform JAXB objects/strings to Java objects and vice versa.

JAXB model modules

specs-xmldsig

W3C XSD schema for signatures http://www.w3.org/2000/09/xmldsig

specs-xades

ETSI EN 319 132-1 XSD schema for XAdES.

specs-trusted-list

ETSI TS 119 612 XSD schema for parsing Trusted Lists.

specs-validation-report

ETSI TS 119 102-2 XSD schema for the Validation report.

specs-asic-manifest

ETSI EN 319 162 schema for ASiCManifest.

---

dss-policy-jaxb

JAXB model of the validation policy.

dss-diagnostic-jaxb

JAXB model of the diagnostic data.

dss-detailed-report-jaxb

JAXB model of the detailed report.

dss-simple-report-jaxb

JAXB model of the simple report.

dss-simple-certificate-report-jaxb

JAXB model of the simple report for certificates.

Utils modules

dss-utils

API with utility methods for String, Collection, I/O,…

dss-utils-apache-commons

Implementation of dss-utils with Apache Commons libraries.

dss-utils-google-guava

Implementation of dss-utils with Google Guava.

i18n

dss-i18n

a module allowing internationalization of generated reports.

Core modules

dss-model

Data model used in almost every module.

dss-crl-parser

API to validate CRLs and retrieve revocation data

dss-crl-parser-stream

Implementation of dss-crl-parser which streams the CRL.

dss-crl-parser-x509crl

Implementation of dss-crl-parser which uses the java object X509CRL.

dss-spi

Interfaces, util classes to manipulate ASN1, compute digests,…

dss-document

Common module to sign and validate document. This module doen't contain any implementation.

dss-service

Implementations to communicate with online resources (TSP, CRL, OCSP).

dss-token

Token definitions and implementations for MS CAPI, PKCS#11, PKCS#12.

validation-policy

Business of the signature's validation (ETSI EN 319 102 / TS 119 172-4).

dss-xades

Implementation of the XAdES signature, extension and validation.

dss-cades

Implementation of the CAdES signature, extension and validation.

dss-pades

Common code which is shared between dss-pades-pdfbox and dss-pades-openpdf.

dss-pades-pdfbox

Implementation of the PAdES signature, extension and validation with PDFBox.

dss-pades-openpdf

Implementation of the PAdES signature, extension and validation with OpenPDF (fork of iText).

dss-asic-common

Common code which is shared between dss-asic-xades and dss-asic-cades.

dss-asic-cades

Implementation of the ASiC-S and ASiC-E signature, extension and validation based on CAdES signatures.

dss-asic-xades

Implementation of the ASiC-S and ASiC-E signature, extension and validation based on XAdES signatures.

dss-tsl-validation

Module which allows loading / parsing / validating of LOTL and TSLs.

WebServices

dss-common-remote-dto

Common classes between all remote services (REST and SOAP).

dss-common-remote-converter

Classes which convert the DTO to DSS Objects.

---

dss-signature-dto

Data Transfer Objects used for signature creation/extension (REST and SOAP).

dss-signature-remote

Common classes between dss-signature-rest and dss-signature-soap.

dss-signature-rest-client

Client for the REST webservices.

dss-signature-rest

REST webservices to sign (getDataToSign, signDocument methods) and extend a signature.

dss-signature-soap-client

Client for the SOAP webservices.

dss-signature-soap

SOAP webservices to sign (getDataToSign, signDocument methods) and extend a signature.

---

dss-server-signing-dto

Data Transfer Objects used for the server signing module (REST and SOAP).

dss-server-signing-common

Common classes for server signing.

dss-server-signing-rest

REST webservice for server signing.

dss-server-signing-rest-client

REST client for server signing (sign method).

dss-server-signing-soap

SOAP webservice for server signing.

dss-server-signing-soap-client

SOAP client for server signing (sign method).

---

dss-validation-dto

Data Transfer Objects used for signature validation (REST and SOAP).

dss-validation-common

Common classes between dss-validation-rest and dss-validation-soap.

dss-validation-rest-client

Client for the REST signature-validation webservices.

dss-validation-soap-client

Client for the SOAP signature-validation webservices.

dss-validation-rest

REST webservices to validate a signature.

dss-validation-soap

SOAP webservices to validate a signature.

---

dss-certificate-validation-dto

Data Transfer Objects used for certificate validation (REST and SOAP).

dss-certificate-validation-common

Common classes between dss-certificate-validation-rest and dss-certificate-validation-soap.

dss-certificate-validation-rest-client

Client for the REST certificate-validation webservice.

dss-certificate-validation-soap-client

Client for the SOAP certificate-validation webservice.

dss-certificate-validation-rest

REST webservice to validate a certificate.

dss-certificate-validation-soap

SOAP webservice to validate a certificate.

---

dss-timestamp-dto

Data Transfer Objects used for timestamp creation.

dss-timestamp-remote-common

Common classes between dss-timestamp-remote-rest and dss-timestamp-remote-soap.

dss-timestamp-remote-rest-client

Client for the REST timestamp webservice.

dss-timestamp-remote-soap-client

Client for the SOAP timestamp webservice.

dss-timestamp-remote-rest

REST webservice to create a timestamp.

dss-timestamp-remote-soap

SOAP webservice to create a timestamp.

Other modules

dss-test

Mocks and util classes for unit tests.

dss-cookbook

Samples and documentation of DSS used to generate this documentation.

DSS Utils

The module dss-utils offers an interface with utility methods to operate on String, Collection, I/O,... DSS framework provides two different implementations with the same behaviour :

- dss-utils-apache-commons : this module uses Apache Commons libraries (commons-lang3, commons-collection4, commons-io and commons-codec).
- dss-utils-google-guava : this module only requires Google Guava (recommended on Android).

If your integration include dss-utils, you will need to select an implementation.

DSS CRL Parser

DSS contains two ways to parse/validate a CRL and to retrieve revocation data. An alternative to the X509CRL java object was developed to face memory issues in case of large CRLs. The X509CRL object fully loads the CRL in memory and can cause OutOfMemoryError.

- dss-crl-parser-x509crl : this module uses the X509CRL java object.
- dss-crl-parser-streams : this module offers an alternative with a CRL streaming (experimental).

If your integration require dss-crl-parser, you will need to choose your implementation.

DSS PAdES

Since the version 5.4, DSS allows generation/extension/validation PAdES signatures with two different frameworks : PDFBox and OpenPDF (fork of iText). The dss-pades module only contains the common code and requires an underlying implementation :

- dss-pades-pdfbox : Supports drawing of custom text, images, as well as text+image, in a signature field.
- dss-pades-openpdf : Supports drawing of custom text OR images in a signature field.

DSS permits to override the visible signature generation with these interfaces :

- eu.europa.esig.dss.pdf.IPdfObjFactory
- eu.europa.esig.dss.pdf.visible.SignatureDrawerFactory (selects the SignatureDrawer depending on the SignatureImageParameters content)
- eu.europa.esig.dss.pdf.visible.SignatureDrawer

A new instance of the IPdfObjFactory can be created with its own SignatureDrawerFactory and injected in the padesService.setPdfObjFactory(IPdfObjFactory). By default, DSS uses an instance of ServiceLoaderPdfObjFactory. This instance checks for any registered implementation in the classpath with the ServiceLoader (potentially a service from dss-pades-pdfbox, dss-pades-openpdf or your own(s)).

DSS PDFBox

Since the version 5.5, DSS allows switching between two implementations of the framework PDFBox : default (original) and native.

- Default Drawer : The original drawer implemented on the PDFBox framework, supports displaying of custom text, images, text+image combination in a signature field. The implementation does not include the provided custom text to the inner PDF structure, instead of it, the drawer creates an image representation of the provided text, which is added to the signature field (i.e. the text is not selectable and not searchable).
- Native Drawer : Since the version 5.5, DSS includes a new implementation of PDFBox Drawer, that allows a user to add a real custom text, image or combination of text and image to a visible signature field. The native implementation embeds the provided custom text to the inner PDF structure, that makes the text selectable and searchable, and also clearer and smoother in comparison with the original implementation.

By default DSS uses "Default Drawer" as the PDFBox implementation. In order to switch the implementation, that allowed in runtime, you have to set a new instance for PdfObjFactory as following:

Runtime PDF Object Factory changing

```
service.setPdfObjFactory(new PdfBoxNativeObjectFactory());
```

Available demonstrations

With the framework, some demonstrations are provided.

dss-mock-tsa	The class which generate false timestamps from a self-signed certificate.
sscd-mocca-adapter	Adapter for the MOCCA connection.
dss-standalone-app	Standalone application which allows signing a document with different formats and tokens (JavaFX).
dss-standalone-app-package	Packaging module for dss-standalone-app.
dss-demo-webapp	Demonstration web application which presents a part of the DSS possibilities.
dss-demo-bundle	Packaging module for dss-demo-webapp.

The demonstrations use a simulated timestamp service (Mock) so that is not recommended for a production usage.

Signature’s profile simplification

The different formats of the digital signature make possible to cover a wide range of real live cases of use of this technique. Thus we distinguish the following formats: XAdES, CAdES, PAdES and ASIC. To each one of them a specific standard is dedicated. The wide variety of options, settings and versions of the standards makes their interoperability very difficult. This is the main reason for which new standards commonly called "baseline profiles" were published. Their goal is to limit the number of options and variants thereby making possible a better interoperability between different actors.

In general can be said that for each format of the digital signature the number of security levels defined in the new standards has been reduced. Below is a comparative table of old and new levels for each format of the signature:

Table 3. Signature supported profiles

XAdES		CAdES		PAdES	
STANDARD	BASELINE	STANDARD	BASELINE	STANDARD	BASELINE
XAdES-BES	XAdES-B	CAdES-BES	CAdES-B	PAdES-BES	PAdES-B
XAdES-EPES		CAdES-EPES		PAdES-EPES	
XAdES-T	XAdES-T	CAdES-T	CAdES-T	PAdES-T	PAdES-T
XAdES-XL	XAdES-LT	CAdES-XL	CAdES-LT	PAdES-XL	PAdES-LT

XAdES-A	XAdES-LTA	CAdES-A	CAdES-LTA	PAdES-LTV	PAdES-LTA

Note that the new version (v4) of the DSS framework is compatible with the baseline profiles, it is no longer possible to use the standard profiles for signing purpose. The validation of the signature still takes into account the old profiles.

Signature profile guide

Below you can find a table specifying various signature possibilities with available in DSS signature's profiles/formats. The vertical column specifies available signature profiles and their extensions. The horizontal row specifies types of documents to be signed with the formats.

Table 4. File formats and Signature types conformance

Signature profiles			XML	PDF	Binary	Digest	Multi files	Parallel signatures
XAdES	Enveloping	Base64 encoded	✓	✓	✓	✗	✓	✓
		Embed XML	✓	✗	✗	✗	XML only	✓
		Manifest	✓	✓	✓	✓	✓	✓
		Canonicalization	✓	✗	✗	✗	XML only	✓
	Enveloped	enveloped transformation	✓	✗	✗	✗	✗	✗
		based on XPath	✓	✗	✗	✗	✗	✓
		based on Filter2	✓	✗	✗	✗	✗	✓
		Canonicalization	✓	✗	✗	✗	XML only	✓
	Detached		✓	✓	✓	✓	✓	✓
	Internally Detached		✓	✗	✗	✗	XML only	✓
CAdES	Enveloping		✓	✓	✓	✗	✗	✓
	Detached		✓	✓	✓	✓	✗	✓
PAdES	Enveloped		✗	✓	✗	✗	✗	✓

ASiC	ASiCS	CAdES/XAdES	✓	✓	✓	✗	✓	✓
	ASiCE	CAdES/XAdES	✓	✓	✓	✗	✓	✓

The XML Signature (XAdES)

The simplest way to address the digital signature passes through the XAdES format. Indeed, it allows visualization of the signature content with a simple text editor. Thus it becomes much easier to make the connection between theoretical concepts and their implementation. Before embarking on the use of the DSS framework, it is advisable to read the following documents:

- XAdES Specifications (cf. [R01])

After reading these documents, it is clear that:

- To electronically sign a document, a signing certificate (that proves the signer's identity) and the access to its associated private key is needed.
- To electronically validate a signed document the signer's certificate containing the public key is needed. To give a more colourful example: when a digitally signed document is sent to a given person or organization in order to be validated, the certificate with the public key used to create the signature must also be provided.

XAdES Profiles

The new ETSI standard defines four conformance levels to address the growing need to protect the validity of the signature in time. Henceforth to denote the level of the signature the word "level" will be used. Follows the list of levels defined in the standard:

- XAdES-BASELINE-**B**: *Basic Electronic Signature* The lowest and simplest version just containing the SignedInfo, SignatureValue, KeyInfo and SignedProperties. This level combines the old -BES and -EPES levels. This form extends the definition of an electronic signature to conform to the identified signature policy.
- XAdES-BASELINE-**T**: *Signature with a timestamp* A timestamp regarding the time of signing is added to protect against repudiation.
- XAdES-BASELINE-**LT**: *Signature with Long Term Data* Certificates and revocation data are embedded to allow verification in future even if their original source is not available. This level is equivalent to the old -XL level.
- XAdES-BASELINE-**LTA**: *Signature with Long Term Data and Archive timestamp* By using periodical timestamping (e.g. each year) compromising is prevented which could be caused by weakening previous signatures during a long-time storage period. This level is equivalent to the old -A level.

Old levels: -BES, -EPES, -C, -X, -XL, -A are not supported any more when signing.

XAdES-BASELINE-B

To start, let's take a simple XML document:

xml_example.xml

```
<?xml version="1.0"?>
<test>Hello World !</test>
```

Since this is an XML document, we will use the XAdES signature and more particularly XAdES-BASELINE-B level, which is the lowest level of protection: just satisfying Directive (cf. [R06]) legal requirements for advanced signature. The normal process of signing wants to sign first with the level -B or level-T, and then later when it becomes necessary to complete the signature with superior levels. However, the framework allows signing directly with any level. When signing data, the resulting signature needs to be linked with the data to which it applies. This can be done either by creating a data set which combines the signature and the data (e.g. by enveloping the data with the signature or including a signature element in the data set) or placing the signature in a separate resource and having some external means for associating the signature with the data. So, we need to define the packaging of the signature, namely ENVELOPED, ENVELOPING, DETACHED or INTERNALLY-DETACHED. More information about supported reference transformations for each signature packaging (except 'Detached'), can be found in the section Reference Transformations

- **ENVELOPED :** when the signature applies to data that surround the rest of the document;
- **ENVELOPING :** when the signed data form a sub-element of the signature itself;
  - Base64 encoded binaries;
  - Embed XML object(s);
  - Embed Manifest object(s).
- **DETACHED :** when the signature relates to the external resource(s) separated from it.
- **INTERNALLY-DETACHED :** when the signature and the related signed data are both included in a parent element (only XML).

For our example, we will use ENVELOPED packaging.

The DSS framework uses 3 atomic steps to sign a document :

1. Compute the digest to be signed;
2. Sign the digest;
3. Sign the document (add the signed digest).

The DSS fully manages the steps 1 and 3. We need to specify how to do the signature operation. DSS offers some implementations in the dss-token module

To write our Java code, we still need to specify the type of KeyStore to use for signing our document, more simply, where the private key can be found. In the package "eu.europa.esig.dss.token", we can choose between different connection tokens :

- **Pkcs11SignatureToken :** allows communicating with SmartCards with the PKCS#11 interface. It requires some installed drivers (dll, sso,...) .
- **Pkcs12SignatureToken :** allows signing with a PKC#12 keystore (.p12 file).

- **MSCAPISignatureToken :** handles the signature with MS CAPI (the Microsoft interface to communicate with SmartCards).
- **JKSSignatureToken :** allows signing with a Java Key Store (.jks file).

The DSS also provides the support for MOCCA framework to communicate with the Smartcard with PC/SC, but it involves the installation of the MOCCA and IAIK libraries.

To know more about the use of the different signature tokens, please consult "Management of Signature Tokens" chapter.

In our example the class: "Pkcs12SignatureToken" will be used. A file in PKCS#12 format must be provided to the constructor of the class. It contains an X.509 private key accompanying the public key certificate and protected by symmetrical password. The certification chain can also be included in this file. It is possible to generate dummy certificates and their chains with OpenSSL. Please visit http://www.openssl.org/ for more details.

This is the complete code that allows you to sign our XML document.

Create a XAdES signature

```
// Preparing parameters for the XAdES signature
XAdESSignatureParameters parameters = new XAdESSignatureParameters();
// We choose the level of the signature (-B, -T, -LT, -LTA).
parameters.setSignatureLevel(SignatureLevel.XAdES_BASELINE_B);
// We choose the type of the signature packaging (ENVELOPED, ENVELOPING, DETACHED).
parameters.setSignaturePackaging(SignaturePackaging.ENVELOPED);
// We set the digest algorithm to use with the signature algorithm. You must use the
// same parameter when you invoke the method sign on the token. The default value is
SHA256
parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// We set the signing certificate
parameters.setSigningCertificate(privateKey.getCertificate());
// We set the certificate chain
parameters.setCertificateChain(privateKey.getCertificateChain());

// Create common certificate verifier
CommonCertificateVerifier commonCertificateVerifier = new CommonCertificateVerifier();

// Create XAdES service for signature
XAdESService service = new XAdESService(commonCertificateVerifier);

// Get the SignedInfo XML segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);

// This function obtains the signature value for signed information using the
// private key and specified algorithm
SignatureValue signatureValue = signingToken.sign(dataToSign, parameters
.getDigestAlgorithm(), privateKey);

// We invoke the service to sign the document with the signature value obtained in
// the previous step.
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

What you may notice is that to sign a document we need to:

- Create an object based on SignatureParameters class. The number of specified parameters depends on the type of signature. Generally, the number of specified parameters depends on the profile of signature. This object also defines some default parameters.
- Choose the profile, packaging, signature digest algorithm.
- Indicate the private key entry to be used.
- Instantiate the adequate signature service.
- Carry out the signature process.

The encryption algorithm is determined by the private key and therefore cannot be compelled by

the setter of the signature parameters object. It will cause an inconsistency in the signature making its validation impossible. This setter can be used in a particular context where the signing process is distributed on different machines and the private key is known only to the signature value creation process. See clause "Signing process" for more information. In the case where the private key entry object is not available, it is possible to choose the signing certificate and its certificate chain as in the following example:

```
// We set the signing certificate
parameters.setSigningCertificate(certificateToken);
// We set the certificate chain
parameters.setCertificateChain(certificateChain);
```

Integrating the certificate chain in the signature simplifies the build of a prospective certificate chain during the validation process.

By default the framework uses the current date time to set the signing date, but in the case where it is necessary to indicate the different time it is possible to use the setter "setSigningDate(Date)" as in the example:

```
// We set the date of the signature.
parameters.bLevel().setSigningDate(new Date());
```

When the specific service is instantiated a certificate verifier must be set. This object is used to provide four different sources of information:

- the source of trusted certificates (based on the trusted list(s) specific to the context);
- the source of intermediate certificates used to build the certificate chain till the trust anchor. This source is only needed when these certificates are not included in the signature itself;
- the source of OCSP;
- the source of CRL.

In the current implementation this object is only used when profile -LT or -LTA are created.

Signing process

Once the parameters of the signature were identified the service object itself must be created. The service used will depend on the type of document to sign. In our case it is an XML file, so we will instantiate a XAdES service. The process of signing takes place in three stages. The first is the `getDataToSign()` method call, passing as a parameter the document to be signed and the previously selected settings. This step returns the data which is going to be digested and encrypted. In our case it corresponds to the SignedInfo XMLDSig element.

```
// Create XAdES service for signature
XAdESService service = new XAdESService(commonCertificateVerifier);

// Get the SignedInfo XML segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);
```

The next step is a call to the function `sign()` which is invoked on the object token representing the KeyStore and not on the service. This method takes three parameters. The first is the array of bytes that must be signed. It is obtained by the previous method invocation. The second is the algorithm used to create the digest. You have the choice between SHA1, SHA256, and SHA512 (this list is not exhaustive). And the last one is the private key entry.

```
DigestAlgorithm digestAlgorithm = parameters.getDigestAlgorithm();
SignatureValue signatureValue = signingToken.sign(dataToSign, digestAlgorithm,
privateKey);
```

The last step of this process is the integration of the signature value in the signature and linking of that one to the signed document based on the selected packaging method. This is the method `signDocument()` on the service. We must pass to it three parameters: again the document to sign, the signature parameters and the value of the signature obtained in the previous step.

This separation into three steps allows use cases where different environments have their precise responsibilities: specifically the distinction between communicating with the token and executing the business logic.

When the breakdown of this process is not necessary, than a simple call to only one method can be done as in the following example:

```
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

Additional attributes

For this type (XAdES-BASELINE-B) of signature it is possible to identify some additional attributes.

XAdES signature with additional signed attributes

```
XAdESSignatureParameters parameters = new XAdESSignatureParameters();

// Basic signature configuration
parameters.setSignaturePackaging(SignaturePackaging.ENVELOPED);
parameters.setSignatureLevel(SignatureLevel.XAdES_BASELINE_B);
parameters.setDigestAlgorithm(DigestAlgorithm.SHA512);
parameters.setSigningCertificate(privateKey.getCertificate());
parameters.setCertificateChain(privateKey.getCertificateChain());

// Configuration of several signed attributes like ...
BLevelParameters bLevelParameters = parameters.bLevel();

// Contains claimed roles assumed by the signer when creating the signature
bLevelParameters.setClaimedSignerRoles(Arrays.asList("Manager"));

// signer location
SignerLocation signerLocation = new SignerLocation();
signerLocation.setCountry("BE");
signerLocation.setStateOrProvince("Luxembourg");
signerLocation.setPostalCode("1234");
signerLocation.setLocality("SimCity");
// Contains the indication of the purported place where the signer claims to have
produced the signature
bLevelParameters.setSignerLocation(signerLocation);

// Identifies the commitment undertaken by the signer in signing (a) signed data
object(s)
// in the context of the selected signature policy
List<String> commitmentTypeIndications = new ArrayList<String>();
commitmentTypeIndications.add(CommitmentType.ProofOfOrigin.getUri());
commitmentTypeIndications.add(CommitmentType.ProofOfApproval.getUri());
bLevelParameters.setCommitmentTypeIndications(commitmentTypeIndications);

CommonCertificateVerifier verifier = new CommonCertificateVerifier();
XAdESService service = new XAdESService(verifier);
service.setTspSource(getOnlineTSPSource());

// Allows setting of content-timestamp (part of the signed attributes)
TimestampToken contentTimestamp = service.getContentTimestamp(toSignDocument,
parameters);
parameters.setContentTimestamps(Arrays.asList(contentTimestamp));

// Signature process with its 3 stateless steps
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);
SignatureValue signatureValue = signingToken.sign(dataToSign, parameters
.getDigestAlgorithm(), privateKey);
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

In XAdES format the following types of a Content Timestamp can be used:

- AllDataObjectsTimeStamp - each time-stamp token within this property covers the full set of references defined in the Signature's SignedInfo element, excluding references of type "SignedProperties".
- IndividualDataObjectsTimeStamp - each time-stamp token within this property covers selected signed data objects.

The code above produces the following signature :

XAdES signature example

```
<xades:SignedProperties Id="xades-id-ea3e16770317bb1a3e97244292931644">
    <xades:SignedSignatureProperties>
        <xades:SigningTime>2018-03-20T08:17:35Z</xades:SigningTime>
        <xades:SigningCertificateV2>
            <xades:Cert>
                <xades:CertDigest>
                    <ds:DigestMethod Algorithm="
http://www.w3.org/2000/09/xmldsig#sha1" />
                    <ds:DigestValue>2FeANjXziO9x2877SfclRlRVjlE=</ds:DigestValue>
                </xades:CertDigest>

<xades:IssuerSerialV2>MD4wNKQyMDAxGzAZBgNVBAMMElJvb3RTZWxmU2lnbmVkRmFrZTERMA8GA1UECgwI
RFNTLXRlc3QCBi7WFNe7Vw==</xades:IssuerSerialV2>
            </xades:Cert>
        </xades:SigningCertificateV2>
        <xades:SignatureProductionPlaceV2>
            <xades:City>SimCity</xades:City>
            <xades:StateOrProvince>Luxembourg</xades:StateOrProvince>
            <xades:PostalCode>1234</xades:PostalCode>
            <xades:CountryName>BE</xades:CountryName>
        </xades:SignatureProductionPlaceV2>
        <xades:SignerRoleV2>
            <xades:ClaimedRoles>
                <xades:ClaimedRole>Manager</xades:ClaimedRole>
            </xades:ClaimedRoles>
        </xades:SignerRoleV2>
    </xades:SignedSignatureProperties>
    <xades:SignedDataObjectProperties>
        <xades:DataObjectFormat ObjectReference="#r-id-1">
            <xades:MimeType>text/xml</xades:MimeType>
        </xades:DataObjectFormat>
        <xades:CommitmentTypeIndication>
            <xades:CommitmentTypeId>
                <xades:Identifier>
http://uri.etsi.org/01903/v1.2.2#ProofOfOrigin</xades:Identifier>
            </xades:CommitmentTypeId>
            <xades:AllSignedDataObjects />
        </xades:CommitmentTypeIndication>
        <xades:CommitmentTypeIndication>
```

```
            <xades:CommitmentTypeId>
                <xades:Identifier>
http://uri.etsi.org/01903/v1.2.2#ProofOfApproval</xades:Identifier>
            </xades:CommitmentTypeId>
            <xades:AllSignedDataObjects />
        </xades:CommitmentTypeIndication>
        <xades:AllDataObjectsTimeStamp Id="TS-
678B5861DBA1469B3AA3DD49DD54D7046BADA578C5561F8ABDA935CE0825279E">
            <ds:CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            <xades:EncapsulatedTimeStamp>
MIAGCSqGSIb3DQEHAq...aAAAAAA=</xades:EncapsulatedTimeStamp>
        </xades:AllDataObjectsTimeStamp>
    </xades:SignedDataObjectProperties>
</xades:SignedProperties>
```

Handling signature policy

With the new standards the policy handling is linked to -B level. The old -EPES level is not used anymore by the framework. This does not alter the structure of the old signature but only modifies how to control the process of its creation.

The DSS framework allows you to reference a signature policy, which is a set of rules for the creation and validation of an electronic signature. It includes two kinds of text:

- In a human readable form: It can be assessed to meet the requirements of the legal and contractual context in which it is being applied.
- In a machine processable form: To facilitate its automatic processing using the electronic rules.

If no signature policy is identified then the signature may be assumed to have been generated or verified without any policy constraints, and hence may be given no specific legal or contractual significance through the context of a signature policy.

The signer may reference the policy either implicitly or explicitly. An implied policy means the signer follows the rules of the policy but the signature does not indicate which policy. It is assumed the choice of policy is clear from the context in which the signature is used and SignaturePolicyIdentifier element will be empty. When the policy is not implied, the signature contains an ObjectIdentier that uniquely identifies the version of the policy in use. The signature also contains a hash of the policy document to make sure that the signer and verifier agree on the contents of the policy document.

This example demonstrates an implicit policy identifier. To implement this alternative you must set SignaturePolicyId to empty string.

XAdES with implicit policy

```
XAdESSignatureParameters parameters = new XAdESSignatureParameters();
parameters.setSignatureLevel(SignatureLevel.XAdES_BASELINE_B);
parameters.setSignaturePackaging(SignaturePackaging.ENVELOPED);
parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// We set the signing certificate
parameters.setSigningCertificate(privateKey.getCertificate());
// We set the certificate chain
parameters.setCertificateChain(privateKey.getCertificateChain());

BLevelParameters bLevelParameters = parameters.bLevel();

Policy policy = new Policy();
policy.setId("");

bLevelParameters.setSignaturePolicy(policy);

// Create common certificate verifier
CommonCertificateVerifier commonCertificateVerifier = new CommonCertificateVerifier();
// Create xadesService for signature
XAdESService service = new XAdESService(commonCertificateVerifier);

// Get the SignedInfo segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);

// This function obtains the signature value for signed information using the
// private key and specified algorithm
DigestAlgorithm digestAlgorithm = parameters.getDigestAlgorithm();
SignatureValue signatureValue = signingToken.sign(dataToSign, digestAlgorithm,
privateKey);

// We invoke the xadesService to sign the document with the signature value obtained
in
// the previous step.
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

An XML segment will be added to the signature's qualified and signed properties:

```
<xades:SignaturePolicyIdentifier>
    <xades:SignaturePolicyImplied/>
</xades:SignaturePolicyIdentifier>
```

The next example demonstrates an explicit policy identifier. This is obtained by setting -B profile signature policy and assigning values to the policy parameters. The Signature Policy Identifier is a URI or OID that uniquely identifies the version of the policy document. The signature will contain the identifier of the hash algorithm and the hash value of the policy document. The DSS framework

does not automatically calculate the hash value; it is to the developer to proceed with the calculation using for example java.security.MessageDigest class (rt.jar). It is important to keep the policy file intact in order to keep the hash constant. It would be wise to make the policy file read-only. See also chapter 7 for further information.

XAdES with explicit policy

```
XAdESSignatureParameters parameters = new XAdESSignatureParameters();
parameters.setSignatureLevel(SignatureLevel.XAdES_BASELINE_B);
parameters.setSignaturePackaging(SignaturePackaging.ENVELOPED);
parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// We set the signing certificate
parameters.setSigningCertificate(privateKey.getCertificate());
// We set the certificate chain
parameters.setCertificateChain(privateKey.getCertificateChain());

BLevelParameters bLevelParameters = parameters.bLevel();

// Get and use the explicit policy
String signaturePolicyId = "http://www.example.com/policy.txt";
DigestAlgorithm signaturePolicyHashAlgo = DigestAlgorithm.SHA256;
String signaturePolicyDescription = "Policy text to digest";
byte[] signaturePolicyDescriptionBytes = signaturePolicyDescription.getBytes();
byte[] digestedBytes = DSSUtils.digest(signaturePolicyHashAlgo,
signaturePolicyDescriptionBytes);

Policy policy = new Policy();
policy.setId(signaturePolicyId);
policy.setDigestAlgorithm(signaturePolicyHashAlgo);
policy.setDigestValue(digestedBytes);

bLevelParameters.setSignaturePolicy(policy);

// Create common certificate verifier
CommonCertificateVerifier commonCertificateVerifier = new CommonCertificateVerifier();
// Create xadesService for signature
XAdESService service = new XAdESService(commonCertificateVerifier);

// Get the SignedInfo segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);

// This function obtains the signature value for signed information using the
// private key and specified algorithm
DigestAlgorithm digestAlgorithm = parameters.getDigestAlgorithm();
SignatureValue signatureValue = signingToken.sign(dataToSign, digestAlgorithm,
privateKey);

// We invoke the xadesService to sign the document with the signature value obtained
in
// the previous step.
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

The following XML segment will be added to the signature qualified & signed properties

(<QualifyingProperties><SignedProperties>):

XAdES Signature Policy element

```
<xades:SignaturePolicyIdentifier>
    <xades:SignaturePolicyId>
        <xades:SigPolicyId>
            <xades:Identifier>http://www.example.com/policy.txt</xades:Identifier>
        </xades:SigPolicyId>
        <xades:SigPolicyHash>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>Uw3PxkrX4SpF03jDvkSu6Zqm9UXDxs56FFXeg7MWy0c=</ds:DigestValue>
        </xades:SigPolicyHash>
    </xades:SignaturePolicyId>
</xades:SignaturePolicyIdentifier>
```

XAdES-BASELINE-T

XAdES-BASELINE-T is a signature for which there exists a trusted time associated to the signature. It provides the initial steps towards providing long term validity and more specifically it provides a protection against repudiation. This extension of the signature can be created as well during the generation process as validation process. However, the case when these validation data are not added during the generation process should no longer occur. The XAdES-BASELINE-T trusted time indications must be created before the signing certificate has been revoked or expired and close to the time that the XAdES signature was produced. The XAdES-BASELINE-T form must be built on a XAdES-BASELINE-B form. The DSS framework allows extending the old -BES and -EPES profiles to the new BASELINE-T profile, indeed there is no difference in the structure of the signature.

To implement this profile of signature you must indicate to the service the TSA source, which delivers from each Timestamp Request a Timestamp Response (RFC 3161 (cf. [R07])) containing tokens. Below is the source code that creates a XAdES-BASELINE-T signature. For our example, we will use the Belgian provider and an instance of OnlineTSPSource (see "TSP Sources" chapter for more details).

Create a XAdES-Baseline-T with an OnlineTSPSource

```
// Preparing parameters for the XAdES signature
XAdESSignatureParameters parameters = new XAdESSignatureParameters();
// We choose the level of the signature (-B, -T, -LT, -LTA).
parameters.setSignatureLevel(SignatureLevel.XAdES_BASELINE_T);
// We choose the type of the signature packaging (ENVELOPED, ENVELOPING, DETACHED).
parameters.setSignaturePackaging(SignaturePackaging.ENVELOPED);
// We set the digest algorithm to use with the signature algorithm. You must use the
// same parameter when you invoke the method sign on the token. The default value is
SHA256
parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// We set the signing certificate
parameters.setSigningCertificate(privateKey.getCertificate());
// We set the certificate chain
parameters.setCertificateChain(privateKey.getCertificateChain());

// Create common certificate verifier
CommonCertificateVerifier commonCertificateVerifier = new CommonCertificateVerifier();
// Create XAdES service for signature
XAdESService service = new XAdESService(commonCertificateVerifier);

// Set the Timestamp source
String tspServer = "http://dss.nowina.lu/pki-factory/tsa/good-tsa";
OnlineTSPSource onlineTSPSource = new OnlineTSPSource(tspServer);
onlineTSPSource.setDataLoader(new TimestampDataLoader()); // uses the specific
content-type
service.setTspSource(onlineTSPSource);

// Get the SignedInfo XML segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);

// This function obtains the signature value for signed information using the
// private key and specified algorithm
SignatureValue signatureValue = signingToken.sign(dataToSign, parameters
.getDigestAlgorithm(), privateKey);

// We invoke the service to sign the document with the signature value obtained in
// the previous step.
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

If the timestamp source is not set a NullPointerException is thrown.

The SignatureTimeStamp mandated by the XAdES-T form appears as an unsigned property within the QualifyingProperties:

XAdES Signature Timestamp

```
<SignatureTimeStamp Id="time-stamp-28a441da-4030-46ef-80e1-041b66c0cb96">
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <EncapsulatedTimeStamp
        Id="time-stamp-token-76234ed8-cc15-46fc-aa95-9460dd601cad">
            MIAGCSqGSIb3DQEHAqCAMIACAQMxCzAJBgUrDgMCGg
            UAMIAGCyqGSIb3DQEJEAEEoIAkgARMMEoCAQEGBoIS
            ...
    </EncapsulatedTimeStamp>
</SignatureTimeStamp>
```

XAdES-BASELINE-LT

This level has to prove that the certification path was valid, at the time of the validation of the signature, up to a trust point according to the naming constraints and the certificate policy constraints from the "Signature Validation Policy". It will add to the signature the CertificateValues and RevocationValues unsigned properties. The CertificateValues element contains the full set of certificates that have been used to validate the electronic signature, including the signer's certificate. However, it is not necessary to include one of those certificates, if it is already present in the ds:KeyInfo element of the signature. This is like DSS framework behaves. In order to find a list of all the certificates and the list of all revocation data, an automatic process of signature validation is executed. To carry out this process an object called CertificateVerifier must be passed to the service. The implementer must set some of its properties (e.g. a source of trusted certificates). The code below shows how to use the default parameters with this object. Please refer to "The Signature Validation" chapter to have the further information. It also includes an example of how to implement this level of signature:

SignXmlXadesLTTest.java

```
// Preparing parameters for the XAdES signature
XAdESSignatureParameters parameters = new XAdESSignatureParameters();
// We choose the level of the signature (-B, -T, -LT, -LTA).
parameters.setSignatureLevel(SignatureLevel.XAdES_BASELINE_LT);
// We choose the type of the signature packaging (ENVELOPED, ENVELOPING, DETACHED).
parameters.setSignaturePackaging(SignaturePackaging.ENVELOPED);
// We set the digest algorithm to use with the signature algorithm. You must use the
// same parameter when you invoke the method sign on the token. The default value is
SHA256
parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// We set the signing certificate
parameters.setSigningCertificate(privateKey.getCertificate());
// We set the certificate chain
parameters.setCertificateChain(privateKey.getCertificateChain());

// Create common certificate verifier
CommonCertificateVerifier commonCertificateVerifier = new CommonCertificateVerifier();
```

```
CommonsDataLoader commonsHttpDataLoader = new CommonsDataLoader();
OCSPDataLoader ocspDataLoader = new OCSPDataLoader();

KeyStoreCertificateSource keyStoreCertificateSource = new KeyStoreCertificateSource
(new File("src/main/resources/keystore.p12"), "PKCS12",
        "dss-password");

LOTLSource lotlSource = new LOTLSource();
lotlSource.setUrl("https://ec.europa.eu/tools/lotl/eu-lotl.xml");
lotlSource.setCertificateSource(keyStoreCertificateSource);
lotlSource.setPivotSupport(true);

TrustedListsCertificateSource tslCertificateSource = new
TrustedListsCertificateSource();

FileCacheDataLoader onlineFileLoader = new FileCacheDataLoader(commonsHttpDataLoader);

CacheCleaner cacheCleaner = new CacheCleaner();
cacheCleaner.setCleanFileSystem(true);
cacheCleaner.setDSSFileLoader(onlineFileLoader);

TLValidationJob validationJob = new TLValidationJob();
validationJob.setTrustedListCertificateSource(tslCertificateSource);
validationJob.setOnlineDataLoader(onlineFileLoader);
validationJob.setCacheCleaner(cacheCleaner);
validationJob.setListOfTrustedListSources(lotlSource);
validationJob.onlineRefresh();

commonCertificateVerifier.setTrustedCertSource(tslCertificateSource);

OnlineCRLSource onlineCRLSource = new OnlineCRLSource();
onlineCRLSource.setDataLoader(commonsHttpDataLoader);
commonCertificateVerifier.setCrlSource(onlineCRLSource);

OnlineOCSPSource onlineOCSPSource = new OnlineOCSPSource();
onlineOCSPSource.setDataLoader(ocspDataLoader);
commonCertificateVerifier.setOcspSource(onlineOCSPSource);

// For test purpose
// Will request unknown OCSP responder / download untrusted CRL
commonCertificateVerifier.setCheckRevocationForUntrustedChains(true);

// Create XAdES service for signature
XAdESService service = new XAdESService(commonCertificateVerifier);
service.setTspSource(getOnlineTSPSource());

// Get the SignedInfo XML segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);

// This function obtains the signature value for signed information using the
// private key and specified algorithm
```

```
SignatureValue signatureValue = signingToken.sign(dataToSign, parameters
.getDigestAlgorithm(), privateKey);

// We invoke the service to sign the document with the signature value obtained in
// the previous step.
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

The following XML segment will be added to the signature qualified and unsigned properties:

Validation data values

```
<CertificateValues>
    <EncapsulatedX509Certificate>
        MIIFNTCCBB2gAwIBAgIBATANB...
    </EncapsulatedX509Certificate>
    <EncapsulatedX509Certificate>
        MIIFsjCCBJqgAwIBAgIDAMoBM...
    </EncapsulatedX509Certificate>
    <EncapsulatedX509Certificate>
        MIIFRjCCBC6gAwIBAgIBATANB...
    </EncapsulatedX509Certificate>
</CertificateValues>
<RevocationValues>
    <OCSPValues>
        <EncapsulatedOCSPValue>
            MIIGzAoBAKCCBsUwggbBBgkr...
        </EncapsulatedOCSPValue>
    </OCSPValues>
</RevocationValues>
```

The use of online sources can significantly increase the execution time of the signing process. For testing purpose you can create your own source of data.

In last example the CommonsHttpDataLoader is used to provide the communication layer for HTTP protocol. Each source which need to go through the network to retrieve data need to have this component set.

XAdES-BASELINE-LTA

When the cryptographic data becomes weak and the cryptographic functions become vulnerable the auditor should take steps to maintain the validity of the signature. The XAdES-BASELINE-A form uses a simple approach called "archive validation data". It adds additional time-stamps for archiving signatures in a way that they are still protected, but also to be able to prove that the signatures were validated at the time when the used cryptographic algorithms were considered safe. The time-stamping process may be repeated every time the protection used becomes weak. Each time-stamp needs to be affixed before either the signing key or the algorithms used by the TSA are no longer secure. XAdES-A form adds the ArchiveTimestamp element within the UnsignedSignatureProperties and may contain several ArchiveTimestamp elements.

Below is an example of the implementation of this level of signature (but in practice, we will rather extend the signature to this level when there is a risk that the cryptographic functions become vulnerable or when one of certificates arrives to its expiration date):

Signature level setting

```
// Allows to set a final signature level
parameters.setSignatureLevel(SignatureLevel.XAdES_BASELINE_LTA);
```

The following XML segment will be added to the signature qualified and unsigned properties:

XAdES Archive Timestamp

```
<ns4:ArchiveTimeStamp
    Id="time-stamp-22b92602-2670-410e-888f-937c5777c685">
    <ds:CanonicalizationMethod
        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <EncapsulatedTimeStamp
        Id="time-stamp-token-0bd5aaf3-3850-4911-a22d-c98dcaca5cea">MIAGCSqGSDHAqCAM⋯
    </EncapsulatedTimeStamp>
</ns4:ArchiveTimeStamp>
```

Various settings

Versions support

DSS supports the following XAdES formats :

Table 5. Supported XAdES versions

	B-level	T-level	LT-level	LTA-level
XAdES 1.1.1	✓	✓	✓	✗
XAdES 1.2.2	✓	✓	✓	✗
XAdES 1.3.2	✓	✓	✓	✓
XAdES 1.4.1	The format contains qualifying properties for XAdES 1.3.2 LTA level			

The XAdES Profile, as well as a customizable prefixes can be set with following methods :

XAdES formats and prefixes

```
// Allows setting of a XAdES namespace (changes a XAdES format)
// Default : XAdESNamespaces.XADES_132 (produces XAdES 1.3.2)
parameters.setXadesNamespace(XAdESNamespaces.XADES_132);

// Defines an XmlDSig prefix
// Default : XAdESNamespaces.XMLDSIG
parameters.setXmldsigNamespace(new DSSNamespace(XMLSignature.XMLNS, "myPrefix"));

// Defines a XAdES 1.4.1 format prefix
// Default : XAdESNamespaces.XADES_141
parameters.setXades141Namespace(XAdESNamespaces.XADES_141);
```

Reference Transformations

In case of 'Enveloping', 'Enveloped' and 'Internally Detached' signatures, it is possible to apply custom transformations for signing references in order to compute proper digest result. Example of a definition reference transformations, you can find below:

Custom transformations definition

```
// Prepare transformations in the proper order
List<DSSTransform> transforms = new ArrayList<>();
DSSTransform envelopedTransform = new EnvelopedSignatureTransform();
transforms.add(envelopedTransform);
DSSTransform canonicalization = new CanonicalizationTransform(CanonicalizationMethod
.EXCLUSIVE_WITH_COMMENTS);
transforms.add(canonicalization);

// Assign reference to the document
List<DSSReference> references = new ArrayList<>();
DSSReference dssReference = new DSSReference();
dssReference.setContents(toSignDocument);
dssReference.setId("r-" + toSignDocument.getName());
dssReference.setTransforms(transforms);
// set empty URI to cover the whole document
dssReference.setUri("");
dssReference.setDigestMethodAlgorithm(DigestAlgorithm.SHA256);
references.add(dssReference);

// Initialize signature parameters
XAdESSignatureParameters parameters = new XAdESSignatureParameters();
parameters.setSignaturePackaging(SignaturePackaging.ENVELOPED);
parameters.setSignatureLevel(SignatureLevel.XAdES_BASELINE_B);
// set references
parameters.setReferences(references);
```

Current version of DSS supports the following transformations:

- Enveloped - removes the current `Signature` element from the digest calculation of the reference.

Does not support parallel signatures!

```
DSSTransform envelopedTransform = new EnvelopedSignatureTransform();
```

- Canonicalization - any canonicalization algorithm that can be used for 'CanonicalizationMethod' can be used as a transform:

```
DSSTransform canonicalization = new CanonicalizationTransform(CanonicalizationMethod
.EXCLUSIVE_WITH_COMMENTS);
```

- Base64 - the transform is used if application needs to sign a RAW data (binaries, images, audio or other formats). The 'Base64 Transform' is not compatible with following signature parameters:
  - Reference contains more than one transform (must be a sole element of the reference transforms);
  - setEmbedXML(true) - embedded setting cannot be used;
  - setManifestSignature(true) - As is apparent from the previous point, Manifest cannot be used with the Base64 Transform as well since it also must be embedded to the signature.

```
DSSDocument document = new InMemoryDocument("Hello World!".getBytes(), "Hello.txt",
MimeType.BINARY);
List<DSSTransform> transforms = new ArrayList<>();
DSSTransform base64Transform = new Base64Transform();
transforms.add(base64Transform);
```

- XPath - allows signing a custom nodes in a signature or embedded document. DSS contains an additional class `XPathEnvelopedSignatureTransform` allowing to exclude signatures from the digested content (used for Enveloped signatures by default). Additional information about the 'XPath Transform' can be found by the link.

```
List<DSSTransform> transforms = new ArrayList<>();
DSSTransform envelopedTransform = new XPathTransform("not(ancestor-or-
self::ds:Signature)");
transforms.add(envelopedTransform);
```

- XPath-2-Filter - an alternative to 'XPath Transform'. Additional information about the 'XPath2Filter Transform' can be found by the link. DSS contains an additional class `XPath2FilterEnvelopedSignatureTransform` allowing to exclude signatures from the digest calculation.

```
List<DSSTransform> transforms = new ArrayList<>();
DSSTransform envelopedTransform = new XPath2FilterTransform("descendant::ds:Signature
", "subtract");
transforms.add(envelopedTransform);
```

- XSLT Transform - This transform requires a 'org.w3.dom.Document' as an input, compatible with the normative XSLT Specification. Must be a sole transform.

All transformations, except Base64, can be applied only to XML objects.

Trust anchor inclusion policy

It is possible to indicate to the framework if the certificate related to the trust anchor should be included to the signature or not. The setter #setTrustAnchorBPPolicy of the BLevelParameters class should be used for this purpose.

This rule applies as follows: when -B level is constructed the trust anchor is not included, when -LT level is constructed the trust anchor is included.

When trust anchor baseline profile policy is defined only the certificates previous to the trust anchor are included when -B level is constructed.

Multiple signatures

In everyday life, there are many examples where it is necessary to have multiple signatures covering the same document, such as a contract to purchase a vehicle. Independent signatures are parallel signatures where the ordering of the signatures is not important. The computation of these signatures is performed on exactly the same input but using different private keys.

The XML Signature Extension (XAdES)

The -B level contains immutable signed properties. Once this level is created, these properties cannot be changed.

The levels -T/-LT/-LTA add unsigned properties to the signature. This means that the properties of these levels could be added afterwards to any AdES signature. This addition helps to make the signature more resistant to cryptographic attacks on a longer period of time. The extension of the signature is incremental, i.e. when you want to extend the signature to the level -LT the lower level (-T) will also be added. The whole extension process is implemented by reusing components from signature production. To extend a signature we proceed in the same way as in the case of a signature, except that you have to call the function "extendDocument" instead of the "sign" function. Note that when the document is signed with several signatures then they are all extended.

XAdES-BASELINE-T

The XAdES-BASELINE-T trusted time indications have to be created before a certificate has been

revoked or expired and close to the time that the XAdES signature was produced. It provides a protection against repudiation. The framework adds the timestamp only if there is no timestamp or there is one but the creation of a new extension of the level-T is deliberate (using another TSA). It is not possible to extend a signature which already incorporates higher level as -LT or -LTA. In the theory it would be possible to add another -T level when the signature has already reached level -LT but the framework prevents this operation. Note that if the signed document contains multiple signatures, then all the signatures will be extended to level -T. It is also possible to sign a document directly at level -T.

Here is an example of creating an extension of type T:

Extend a XAdES signature

```
DSSDocument document = new FileDocument("src/test/resources/signedXmlXadesB.xml");

XAdESSignatureParameters parameters = new XAdESSignatureParameters();
parameters.setSignatureLevel(SignatureLevel.XAdES_BASELINE_T);

CommonCertificateVerifier certificateVerifier = new CommonCertificateVerifier();
XAdESService xadesService = new XAdESService(certificateVerifier);
xadesService.setTspSource(getOnlineTSPSource());

DSSDocument extendedDocument = xadesService.extendDocument(document, parameters);
```

Here is the result of adding a new extension of type-T to an already existing -T level signature:

XAdES Unsigned Signature Properties

```
<UnsignedSignatureProperties>
    <SignatureTimeStamp Id="time-stamp-b16a2552-b218-4231-8982-40057525fbb5">
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
/>
        <EncapsulatedTimeStamp Id="time-stamp-token-39fbf78c-9cec-4cc1-ac21-
a467d2238405">          MIAGCSqGSIb3DQEHAq...
        </EncapsulatedTimeStamp>
    </SignatureTimeStamp>
    <SignatureTimeStamp Id="time-stamp-5ffab0d9-863b-414a-9690-a311d3e1af1d">
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
/>
        <EncapsulatedTimeStamp Id="time-stamp-token-87e8c599-89e5-4fb3-a32a-
e5e2a40073ad">          MIAGCSqGSIb3DQEHAq...
        </EncapsulatedTimeStamp>
    </SignatureTimeStamp>
</UnsignedSignatureProperties>
```

XAdES-BASELINE-LT and -LTA

For these types of extensions, the procedure to follow is the same as the case of the extension of type T. Please refer to the chapter XAdES Profiles (XAdES) to know specific parameters for each

level of signature and which must be positioned.

XAdES and specific schema version

Some signatures may have been created with an older version of XAdES standard using different schema definition. To take into account the validation of such signatures the class eu.europa.esig.dss.xades.validation.XPathQueryHolder was created. This class includes all XPath queries which are used to explore the elements of the signature. It is now easy to extend this class in order to define specific queries to a given schema. The DSS framework proposes in standard the class eu.europa.esig.dss.xades.validation.XAdES111XPathQueryHolder that defines the XPath queries for the version "http://uri.etsi.org/01903/v1.1.1#" of XAdES standard.

When carrying out the validation process of the signature, the choice of query holder to be used is taken by invoking the method: eu.europa.esig.dss.xades.validation.XPathQueryHolder#canUseThisXPathQueryHolder

This choice is made based on the namespace. If the namespace is: http://uri.etsi.org/01903/v1.3.2# then the default query holder is used, if the namespace is http://uri.etsi.org/01903/v1.1.1# the XAdES111XPathQueryHolder is used. The element used to choose the namespace is "QualifyingProperties".

To implement another query holder the class XPathQueryHolder must be extended, new XPath queries defined and the method canUseThisXPathQueryHolder overridden.

In case there is a need to use only a specific query holder the following steps should be followed:

- Call: eu.europa.esig.dss.xades.validation.XMLDocumentValidator#clearQueryHolders
- Call: eu.europa.esig.dss.xades.validation.XMLDocumentValidator#addXPathQueryHolder and pass the specific query holder

The signature validation

Generally and following ETSI standard, the validation process of an electronic signature must provide one of these three following statuses: TOTAL-FAILED, TOTAL-PASSED or INDETERMINATE. A TOTAL-PASSED response indicates that the signature has passed verification and it complies with the signature validation policy. A TOTAL_FAILED response indicates that either the signature format is incorrect or that the digital signature value fails the verification. An INDETERMINATE validation response indicates that the format and digital signature verifications have not failed but there is an insufficient information to determine if the electronic signature is valid. For each of the validation checks, the validation process must provide information justifying the reasons for the resulting status indication as a result of the check against the applicable constraints. In addition, the ETSI standard defines a consistent and accurate way for justifying statuses under a set of sub-indications.

Validation Process

Since version 4.7 of the DSS framework the validation process is based on the latest ETSI standard

[R08]. It is driven by the validation policy and allows long term signature validation. It not only verifies the existence of certain data and their validity, but it also checks the temporal dependences between these elements. The signature check is done following basic building blocks. On the simplified diagram below, showing the process of the signature validation, you can follow the relationships between each building block which represents a logic set of checks used in validation process.

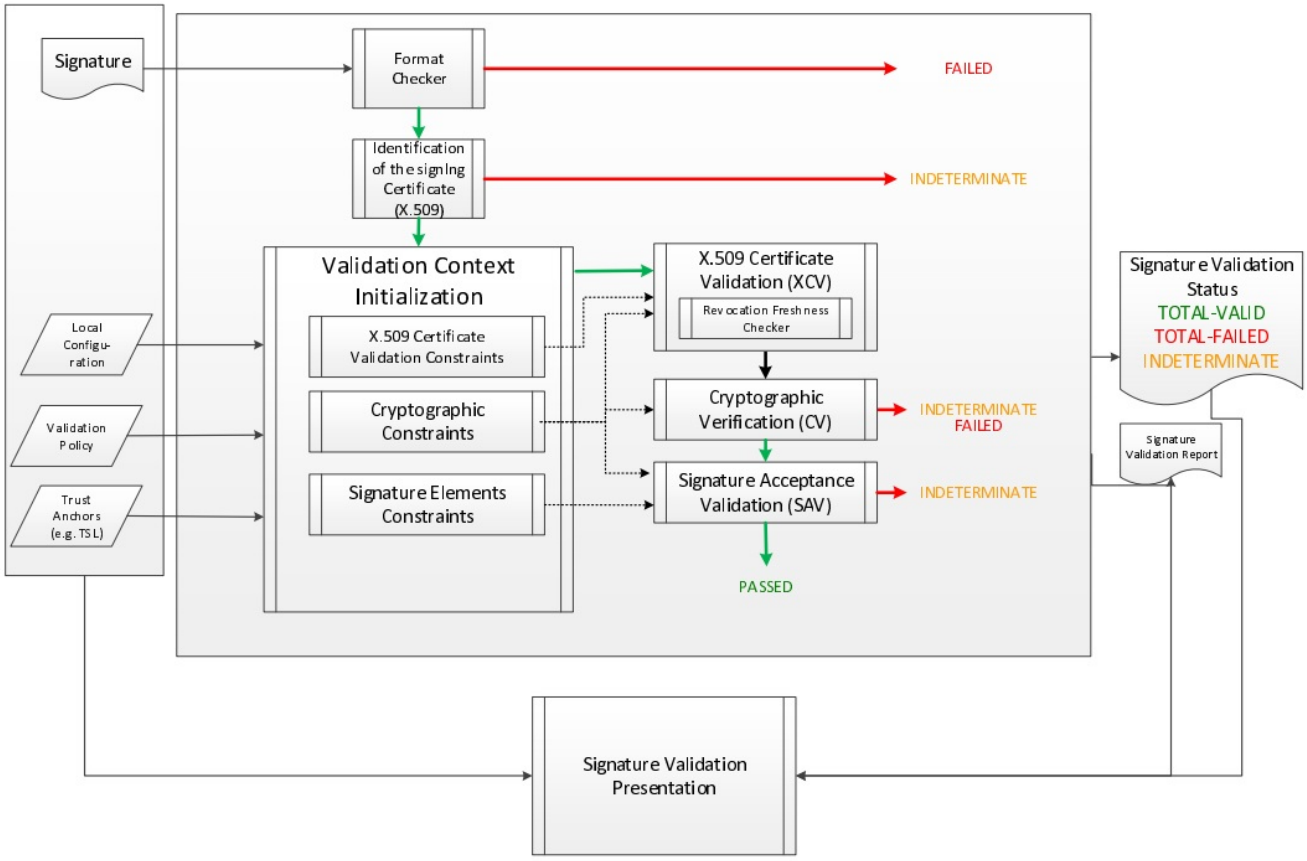


Figure 1. Signature Validation Process

Note that the current version of the framework during the validation process does not indicate what part of a document was signed. However, in a case of XAdES signature XPath transformations presented in the signature will be applied, in the case of CAdES or PAdES signature the whole document must be signed.

At the end of the validation process four reports are created. They contain the different detail levels concerning the validation result. They provide four kinds of visions for the validation process: macroscopic, microscopic, input data and ETSI Validation report conformant with the standard [R08]. For more information about these reports, please refer to "Simple Report" chapter.

Below is the simplest example of the validation of the signature of a document. The first thing to do is instantiating an object named validator, which orchestrates the verification of the different rules. To perform this it is necessary to invoke a static method fromDocument() on the abstract class `SignedDocumentValidator`. This method returns the object in question whose type is chosen dynamically based on the type of source document.

The next step is to create an object that will check the status of a certificate using the Trusted List model (see "Trusted Lists of Certification Service Provider" for more information). In order to achieve this, an instance of a `CertificateVerifier` must be created with a defined source of trusted

certificates. In our example, the trusted source is instantiated with `CommonTrustedCertificateSource` class. As well as a trusted source the CertificateVerifier object needs an OCSP and/or CRL source and a TSL source (which defines how the certificates are retrieved from the Trusted Lists). See chapter "Management of CRL and OCSP Sources" for more information concerning sources.

Validation of a signature

```
// First, we need a Certificate verifier
CertificateVerifier cv = new CommonCertificateVerifier();

// We can inject several sources. eg: OCSP, CRL, AIA, trusted lists

// Capability to download resources from AIA
cv.setDataLoader(new CommonsDataLoader());

// Capability to request OCSP Responders
cv.setOcspSource(new OnlineOCSPSource());

// Capability to download CRL
cv.setCrlSource(new OnlineCRLSource());

// Create an instance of a trusted certificate source
CommonTrustedCertificateSource trustedCertSource = new CommonTrustedCertificateSource
();
// import the keystore as trusted
trustedCertSource.importAsTrusted(keystoreCertSource);

// We now add trust anchors (trusted list, keystore,...)
cv.setTrustedCertSource(trustedCertSource);

// We also can add missing certificates
cv.setAdjunctCertSource(adjunctCertSource);

// Here is the document to be validated (any kind of signature file)
DSSDocument document = new FileDocument(new File(
"src/test/resources/signedXmlXadesLT.xml"));

// We create an instance of DocumentValidator
// It will automatically select the supported validator from the classpath
SignedDocumentValidator documentValidator = SignedDocumentValidator.fromDocument
(document);

// We add the certificate verifier (which allows to verify and trust certificates)
documentValidator.setCertificateVerifier(cv);

// Here, everything is ready. We can execute the validation (for the example, we use
the default and embedded
// validation policy)
Reports reports = documentValidator.validateDocument();

// We have 3 reports
```

```
// The diagnostic data which contains all used and static data
DiagnosticData diagnosticData = reports.getDiagnosticData();

// The detailed report which is the result of the process of the diagnostic data and
the validation policy
DetailedReport detailedReport = reports.getDetailedReport();

// The simple report is a summary of the detailed report (more user-friendly)
SimpleReport simpleReport = reports.getSimpleReport();
```

When using the TrustedListsCertificateSource class, for performance reasons, consider creating a single instance of this class and initialize it only once.

In general, the signature must cover the entire document so that the DSS framework can validate it. However, for example in the case of a XAdES signature, some transformations can be applied on the XML document. They can include operations such as canonicalization, encoding/decoding, XSLT, XPath, XML schema validation, or XInclude. XPath transforms permit the signer to derive an XML document that omits portions of the source document. Consequently those excluded portions can change without affecting signature validity.

SignedDocumentValidator

For execution of the validation process, DSS uses the 'SignedDocumentValidator' class. The DSS framework provides five implementations of validator:

- `XMLDocumentValidator` - validates documents in XML format (XAdES format);
- `CMSDocumentValidator` - validates documents in CMS format (CAdES format);
- `PDFDocumentValidator` - validates documents in PDF format (PADES format);
- `ASiCContainerWithXAdESValidator` - validates ASiC with XAdES containers;
- `ASiCContainerWithCAdESValidator` - validates ASiC with CAdES containers;
- `DetachedTimestampValidator` - validates CMS timestamps provided alone.

DSS initializes a relevant validator based on specific characteristics of an input file (e.g. a PDF file version declaration for a PDF file). It checks the file format and loads the required validator from a classpath. Below you can find a list of settings that can be used for the configuration of the class.

SignedDocumentValidator usage

```
// The method allows instantiation of a related validator for a provided document
// independently on its format (the target dss module must be added as dependency)
SignedDocumentValidator documentValidator = SignedDocumentValidator.fromDocument
(document);

// Allows specifying a custom certificate verifier (online or offline)
documentValidator.setCertificateVerifier(new CommonCertificateVerifier());
```

```
// Allows defining of a signing certificate in the explicit way, in case if the
certificate
// is not provided in the signature itself (can be used for non-ASiC signatures)
documentValidator.defineSigningCertificate(DSSUtils.loadCertificateFromBase64EncodedSt
ring(

"MIIC9TCCAd2gAwIBAgIBAjANBgkqhkiG9w0BAQUFADArMQswCQYDVQQGEwJBQTEMMAoGA1UEChMDRFNTMQ4wD
AYDVQQDEwVJQ0EgQTAeFw0xMzEyMDIxNzMzMTBaFw0xNTEyMDIxNzMzMTBaMDAxCzAJBgNVBAYTAkFBMQwwCgY
DVQQKEwNEU1MxEzARBgNVBAMTCnVzZXIgQSBSU0EwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAJUHHAphm
SDdQ1t62tppK+dLTANsE2nAj+HCpasS3ohlBsrhteRsvTAbrDyIzCmTYWu/nVI4TGvbzBESwV/QitlkoMLpYFw
32MIBf2DLmECzGJ3vm5haw6u8S9quR1h8Vu7QWd+5KMabZuR+j91RiSuoY0xS2ZQxJw1vhvW9hRYjAgMBAAGjg
aIwgZ8wCQYDVR0TBAIwADAdBgNVHQ4EFgQU9ESnTWfwg13c3LQZzqqwibY5WVYwUwYDVR0jBEwwSoAUIO1CDsB
SUcEoFZxKaWf1PAL1U+uhL6QtMCsxDDAKBgNVBAoTA0RTUzELMAkGA1UEBhMCQUExDjAMBgNVBAMTBVJDQSBBg
gEBMAsGA1UdDwQEAwIHgDARBgNVHSAECjAIMAYGBFUdIAAwDQYJKoZIhvcNAQEFBQADggEBAGnhhnoyVUhDnr/
BSbZ/uWfSuwzFPG+2V9K6WxdIaaXOORFGIdFwGlAwA/Qzpq9snfBxuTkAykxq0uEDhHTj0qXxWRjQ+Dop/Drmc
coF/zDvgGusyY1YXaABd/kc3IYt7ns7z3tpiqIz4A7a/UHplBRXfqjyaZurZuJQRaSdxh6CNhdEUiUBxkbb1Sd
MjuOgjzSDjcDjcegjvDquMKdDetvtu2Qh4ConBBo3fUImwiFRWnbudS5H2HE18ikC7gY/QIuNr7USf1PNyUgcG
2g31cMtemj7UTBHZ2V/jPf7ZXqwfnVSaYkNvM3weAI6R3PI0STjdxN6a9qjt9xld40YEdw=="));


// Sets the detached contents that were used for the detached signature creation
documentValidator.setDetachedContents(Arrays.asList(new InMemoryDocument("Hello
world!".getBytes())));

// Allows defining a custom Process Executor
// By default used {@code new DefaultSignatureProcessExecutor()}
documentValidator.setProcessExecutor(new DefaultSignatureProcessExecutor());

// Sets custom Signature Policy Provider
documentValidator.setSignaturePolicyProvider(new SignaturePolicyProvider());

// Sets an expected signature validation level
// The recommended level is ARCHIVAL_DATA (maximal level of the validation)
// Default : ValidationLevel.ARCHIVAL_DATA
documentValidator.setValidationLevel(ValidationLevel.ARCHIVAL_DATA);

// Sets if the ETSI validation report must be created
// If true, it will become accessible through the method below
// Default : true
documentValidator.setEnableEtsiValidationReport(true);

// Executes the validation process and produces validation reports:
// Simple report, Detailed report, Diagnostic data and ETSI Validation Report (if
enabled)
Reports reports = documentValidator.validateDocument();

// Returns ETSI Validation Report (if enabled, NULL otherwise)
ValidationReportType etsiValidationReport = reports.getEtsiValidationReportJaxb();
```

Validation Result Materials

The result of the validation process consists of three elements:

- the Simple Report,
- the Detailed Report,
- the Diagnostic Data and
- the ETSI Validation Report.

All these reports are encoded using XML, which allows the implementer to easily manipulate and extract information for further analysis. For each report, XML Schema and JaxB model are available as maven dependencies.

DSS also provides XSLT to able to generate PDF or HTML reports (simple and detailed reports).

You will find below a detailed description of each of these elements.

Simple Report

This is a sample of the simple validation report:

Simple Report

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SimpleReport xmlns="http://dss.esig.europa.eu/validation/simple-report">
    <ValidationPolicy>
        <PolicyName>QES AdESQC TL based</PolicyName>
        <PolicyDescription>Validate electronic signatures and indicates whether they
are Advanced electronic Signatures (AdES), AdES supported by a Qualified Certificate
(AdES/QC) or a
                Qualified electronic Signature (QES). All certificates and their
related chains supporting the signatures are validated against the EU Member State
Trusted Lists (this includes
                signer's certificate and certificates used to validate certificate
validity status services - CRLs, OCSP, and time-stamps).
        </PolicyDescription>
    </ValidationPolicy>
    <ValidationTime>2020-01-21T06:07:03</ValidationTime>
    <DocumentName>EmptyPage-signed-pades-baseline-b.pdf</DocumentName>
    <ValidSignaturesCount>1</ValidSignaturesCount>
    <SignaturesCount>1</SignaturesCount>
    <Signature SignatureFormat="PAdES-BASELINE-B" Id="S-
3723FC8ECE93FD281E21E7239EFAFA0E286306CB5F57F777F5E3A0A3426CA6B1">
        <CertificateChain>
            <Certificate>
                <id>C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F</id>
                <qualifiedName>Pierrick Vandenbroucke (Signature)</qualifiedName>
            </Certificate>
            <Certificate>
                <id>C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA</id>
                <qualifiedName>Citizen CA</qualifiedName>
            </Certificate>
            <Certificate>
                <id>C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0</id>
                <qualifiedName>Belgium Root CA4</qualifiedName>
            </Certificate>
        </CertificateChain>
        <Indication>TOTAL_PASSED</Indication>
        <Warnings>The organization name is missing in the trusted
certificate!</Warnings>
        <SigningTime>2019-08-27T14:06:11</SigningTime>
        <BestSignatureTime>2020-01-21T06:07:03</BestSignatureTime>
        <SignedBy>Pierrick Vandenbroucke (Signature)</SignedBy>
        <SignatureLevel description="Qualified Electronic Signature">
QESig</SignatureLevel>
        <SignatureScope name="Full PDF" scope="FULL">Full document</SignatureScope>
    </Signature>
</SimpleReport>
```

The result of the validation process is based on very complex rules. The purpose of this report is to make as simple as possible the information while keeping the most important elements. Thus the end user can, at a glance, have a synthetic view of the validation. To build this report the framework uses some simple rules and the detailed report as input.

Detailed Report

This is a sample of the detailed validation report. Its structure is based on the ETSI standard [R08] and is built around Basic Building Blocks, Basic Validation Data, Timestamp Validation Data, AdES-T Validation Data and Long Term Validation Data. Some segments were deleted to make reading easier. They are marked by three dots:

Detailed Report

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DetailedReport xmlns="http://dss.esig.europa.eu/validation/detailed-report">
    <Signature Id="S-3723FC8ECE93FD281E21E7239EFAFA0E286306CB5F57F777F5E3A0A3426CA6B1
">
        <ValidationProcessBasicSignature Title="Validation Process for Basic
Signatures">
            <Constraint Id="S-
3723FC8ECE93FD281E21E7239EFAFA0E286306CB5F57F777F5E3A0A3426CA6B1">
                <Name NameId="ADEST_ROBVPIIC">Is the result of the Basic Validation
Process conclusive?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
            <ProofOfExistence>
                <Time>2020-01-21T06:07:03</Time>
            </ProofOfExistence>
        </ValidationProcessBasicSignature>
        <ValidationProcessLongTermData Title="Validation Process for Signatures with
Time and Signatures with Long-Term Validation Data">
            <Constraint>
                <Name NameId="LTV_ABSV">Is the result of the Basic Validation Process
acceptable?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint Id="R-
5E2868FF9EE4FC069B79171D768B0A90AB137847ADD4F5344EB5F153BB1F19C9">
                <Name NameId="ADEST_RORPIIC">Is the result of the revocation data
validation process acceptable?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_XCV_IRDC">Is the revocation data consistent?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Revocation R-
5E2868FF9EE4FC069B79171D768B0A90AB137847ADD4F5344EB5F153BB1F19C9 thisUpdate 2020-01-21
```

```
06:07 is in the certificate validity range : 2017-01-25 22:12 - 2027-01-21
23:59</AdditionalInfo>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_XCV_IARDPFC">Is an acceptable revocation data
present for the certificate?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Certificate Id = C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F</AdditionalInfo>
            </Constraint>
            <Constraint Id="R-
4B614C6CFB8CF6B1F9E9C74E07464CE6483EA31E0C3D58A7D4C56D540EDF40FB">
                <Name NameId="ADEST_RORPIIC">Is the result of the revocation data
validation process acceptable?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_XCV_IRDC">Is the revocation data consistent?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Revocation R-
4B614C6CFB8CF6B1F9E9C74E07464CE6483EA31E0C3D58A7D4C56D540EDF40FB thisUpdate 2020-01-01
11:00 is in the certificate validity range : 2015-11-25 10:00 - 2027-07-25
10:00</AdditionalInfo>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_XCV_IARDPFC">Is an acceptable revocation data
present for the certificate?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Certificate Id = C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA</AdditionalInfo>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_SAV_ISVA">Is the signature acceptable?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
            <ProofOfExistence>
                <Time>2020-01-21T06:07:03</Time>
            </ProofOfExistence>
        </ValidationProcessLongTermData>
        <ValidationProcessArchivalData Title="Validation Process for Signatures with
Archival Data">
            <Constraint>
                <Name NameId="ARCH_LTVV">Is the result of the LTV validation process
acceptable?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
```

```
            </Conclusion>
            <ProofOfExistence>
                <Time>2020-01-21T06:07:03</Time>
            </ProofOfExistence>
        </ValidationProcessArchivalData>
        <ValidationSignatureQualification Id="S-
3723FC8ECE93FD281E21E7239EFAFA0E286306CB5F57F777F5E3A0A3426CA6B1"
SignatureQualification="QESig" Title="Signature Qualification">
            <Constraint>
                <Name NameId="QUAL_IS_ADES">Is the signature/seal an acceptable AdES
digital signature (ETSI EN 319 102-1)?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="QUAL_CERT_TRUSTED_LIST_REACHED">Has a trusted list been
reached for the certificate chain?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint Id="LOTL-
EC2AE37FE9A43B48B1CFE2A57EBEE2BD6373EDFF36537EB1BC905747ACBF4C3B">
                <Name NameId="QUAL_TRUSTED_LIST_ACCEPT">Is the trusted list
acceptable?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Trusted List : https://ec.europa.eu/tools/lotl/eu-
lotl.xml</AdditionalInfo>
            </Constraint>
            <Constraint Id="TL-
61C0487109BE27255C19CFF26D8F56BEA621E7F381A7B4CBE7FB4750BD477BF9">
                <Name NameId="QUAL_TRUSTED_LIST_ACCEPT">Is the trusted list
acceptable?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Trusted List : https://tsl.belgium.be/tsl-
be.xml</AdditionalInfo>
            </Constraint>
            <Constraint>
                <Name NameId="QUAL_QC_AT_ST">Is the certificate qualified at (best)
signing time?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="QUAL_FOR_SIGN_AT_ST">Is the certificate for eSig at
(best) signing time?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="QUAL_QC_AT_CC">Is the certificate qualified at issuance
time?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="QUAL_QSCD_AT_ST">Does the private key reside in a QSCD
```

```
at (best) signing time?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
            <ValidationCertificateQualification Id="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F" DateTime="2017-01-
25T22:12:12" ValidationTime="CERTIFICATE_ISSUANCE_TIME" CertificateQualification="QC
for eSig with QSCD" Title="Certificate Qualification at certificate issuance time">
                <Constraint>
                    <Name NameId="QUAL_HAS_CAQC">Is the certificate related to a
CA/QC?</Name>
                    <Status>OK</Status>
                </Constraint>
                ...
                <Conclusion>
                    <Indication>PASSED</Indication>
                </Conclusion>
            </ValidationCertificateQualification>
            <ValidationCertificateQualification Id="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F" DateTime="2020-01-
21T06:07:03" ValidationTime="BEST_SIGNATURE_TIME" CertificateQualification="QC for
eSig with QSCD" Title="Certificate Qualification at best signature time">
                <Constraint>
                    <Name NameId="QUAL_HAS_CAQC">Is the certificate related to a
CA/QC?</Name>
                    <Status>OK</Status>
                </Constraint>
                ...
                <Conclusion>
                    <Indication>PASSED</Indication>
                </Conclusion>
            </ValidationCertificateQualification>
        </ValidationSignatureQualification>
    </Signature>
    <BasicBuildingBlocks Id="R-
4B614C6CFB8CF6B1F9E9C74E07464CE6483EA31E0C3D58A7D4C56D540EDF40FB" Type="REVOCATION">
        <ISC Title="Identification of the Signing Certificate">
            ...
        </ISC>
        <XCV Title="X509 Certificate Validation">
            ...
        </XCV>
        <CV Title="Cryptographic Verification">
            ...
        </CV>
        <SAV ValidationTime="2020-01-21T06:07:03" Title="Signature Acceptance
Validation">
            ...
        </SAV>
```

```
        <CertificateChain>
            <ChainItem Id="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
                <Source>TRUSTED_LIST</Source>
            </ChainItem>
        </CertificateChain>
        <Conclusion>
            <Indication>PASSED</Indication>
        </Conclusion>
    </BasicBuildingBlocks>
    <BasicBuildingBlocks Id="R-
5E2868FF9EE4FC069B79171D768B0A90AB137847ADD4F5344EB5F153BB1F19C9" Type="REVOCATION">
        <ISC Title="Identification of the Signing Certificate">
            ...
        </ISC>
        <XCV Title="X509 Certificate Validation">
           ...
        </XCV>
        <CV Title="Cryptographic Verification">
            ...
        </CV>
        <SAV ValidationTime="2020-01-21T06:07:03" Title="Signature Acceptance
Validation">
            ...
        </SAV>
        <Conclusion>
            <Indication>PASSED</Indication>
        </Conclusion>
    </BasicBuildingBlocks>
    <BasicBuildingBlocks Id="S-
3723FC8ECE93FD281E21E7239EFAFA0E286306CB5F57F777F5E3A0A3426CA6B1" Type="SIGNATURE">
        <FC Title="Format Checking">
            <Constraint>
                <Name NameId="BBB_FC_IEFF">Does the signature format correspond to an
expected format?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_FC_ISD">Is the signature identification not
ambiguous?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_FC_IOSIP">Is only one SignerInfo present?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
        </FC>
        <ISC Title="Identification of the Signing Certificate">
```

```
            <Constraint>
                <Name NameId="BBB_ICS_ISCI">Is there an identified candidate for the
signing certificate?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_ICS_ISASCP">Is the signed attribute: signing-
certificate present?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_ICS_ISACDP">Is the signed attribute: cert-digest of
the certificate present?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_ICS_ICDVV">Does the certificate digest value match a
digest value found in the certificate reference(s)?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_ICS_AIDNASNE">Are the issuer distinguished name and
the serial number equal?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
            <CertificateChain>
                <ChainItem Id="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F">
                    <Source>SIGNATURE</Source>
                </ChainItem>
                <ChainItem Id="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA">
                    <Source>SIGNATURE</Source>
                </ChainItem>
                <ChainItem Id="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
                    <Source>TRUSTED_LIST</Source>
                </ChainItem>
            </CertificateChain>
        </ISC>
        <VCI Title="Validation Context Initialization">
            <Constraint>
                <Name NameId="BBB_VCI_ISPK">Is the signature policy known?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
```

```
        </VCI>
        <XCV Title="X509 Certificate Validation">
            <Constraint>
                <Name NameId="BBB_XCV_CCCBB">Can the certificate chain be built till a
trust anchor?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint Id="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F">
                <Name NameId="BBB_XCV_SUB">Is the certificate validation
conclusive?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint Id="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA">
                <Name NameId="BBB_XCV_SUB">Is the certificate validation
conclusive?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint Id="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
                <Name NameId="BBB_XCV_SUB">Is the certificate validation
conclusive?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
            <SubXCV Id="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F" TrustAnchor="false"
Title="Certificate">
                <Constraint>
                    <Name NameId="QUAL_UNIQUE_CERT">Is the certificate unique?</Name>
                    <Status>OK</Status>
                </Constraint>
                <Constraint>
                    <Name NameId="BBB_XCV_PSEUDO_USE">Is a pseudonym used?</Name>
                    <Status>OK</Status>
                </Constraint>
                <Constraint>
                    <Name NameId="BBB_XCV_ISNSSC">Is certificate not self-
signed?</Name>
                    <Status>OK</Status>
                </Constraint>
                ...
                <Constraint>
                    <Name NameId="BBB_XCV_ICTIVRSC">Is the current time in the
validity range of the signers certificate?</Name>
                    <Status>OK</Status>
                    <AdditionalInfo>Certificate validity : 2017-01-25 22:12 to 2027-
01-21 23:59</AdditionalInfo>
```

```
                </Constraint>
                <Conclusion>
                    <Indication>PASSED</Indication>
                </Conclusion>
                <RAC Id="R-
5E2868FF9EE4FC069B79171D768B0A90AB137847ADD4F5344EB5F153BB1F19C9" Title="Revocation
Acceptance Validation">
                    <Constraint>
                        <Name NameId="BBB_XCV_IRDC">Is the revocation data
consistent?</Name>
                        <Status>OK</Status>
                        <AdditionalInfo>Revocation R-
5E2868FF9EE4FC069B79171D768B0A90AB137847ADD4F5344EB5F153BB1F19C9 thisUpdate 2020-01-21
06:07 is in the certificate validity range : 2017-01-25 22:12 - 2027-01-21
23:59</AdditionalInfo>
                    </Constraint>
                    ...
                    <Conclusion>
                        <Indication>PASSED</Indication>
                    </Conclusion>
                    <RevocationProductionDate>2020-01-
21T06:07:03</RevocationProductionDate>
                </RAC>
                <RFC Id="R-
5E2868FF9EE4FC069B79171D768B0A90AB137847ADD4F5344EB5F153BB1F19C9" Title="Revocation
Freshness Checker">
                    <Constraint>
                        <Name NameId="BBB_XCV_IARDPFC">Is an acceptable revocation
data present for the certificate?</Name>
                        <Status>OK</Status>
                    </Constraint>
                    ...
                    <Conclusion>
                        <Indication>PASSED</Indication>
                    </Conclusion>
                </RFC>
            </SubXCV>
            <SubXCV Id="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA" TrustAnchor="false"
Title="Certificate">
                ...
                <Conclusion>
                    <Indication>PASSED</Indication>
                </Conclusion>
            </SubXCV>
            <SubXCV Id="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0" TrustAnchor="true"
Title="Certificate">
                <Conclusion>
                    <Indication>PASSED</Indication>
                </Conclusion>
```

```
            </SubXCV>
        </XCV>
        <CV Title="Cryptographic Verification">
            <Constraint>
                <Name NameId="BBB_CV_IRDOF">Has the reference data object been
found?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Reference : MESSAGE_DIGEST</AdditionalInfo>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_CV_IRDOI">Is the reference data object
intact?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Reference : MESSAGE_DIGEST</AdditionalInfo>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_CV_ISI">Is the signature intact?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Id = S-
3723FC8ECE93FD281E21E7239EFAFA0E286306CB5F57F777F5E3A0A3426CA6B1</AdditionalInfo>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
        </CV>
        <SAV ValidationTime="2020-01-21T06:07:03" Title="Signature Acceptance
Validation">
            <Constraint>
                <Name NameId="BBB_SAV_ISSV">Is the structure of the signature
valid?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_SAV_ISQPSTP">Is the signed qualifying property:
signing-time present?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_SAV_ISQPMDOSPP">Is the signed qualifying property:
message-digest or SignedProperties present?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="ASCCM">Are signature cryptographic constraints
met?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Validation time : 2020-01-21 06:07 for token with ID :
[S-3723FC8ECE93FD281E21E7239EFAFA0E286306CB5F57F777F5E3A0A3426CA6B1]</AdditionalInfo>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
```

```
            </Conclusion>
            <CryptographicInfo>
                <Algorithm>http://www.w3.org/2001/04/xmldsig-more#rsa-
sha256</Algorithm>
                <KeyLength>2048</KeyLength>
                <Secure>true</Secure>
                <NotAfter>2022-12-31T23:00:00</NotAfter>
            </CryptographicInfo>
        </SAV>
        <CertificateChain>
            <ChainItem Id="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F">
                <Source>SIGNATURE</Source>
            </ChainItem>
            <ChainItem Id="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA">
                <Source>SIGNATURE</Source>
            </ChainItem>
            <ChainItem Id="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
                <Source>TRUSTED_LIST</Source>
            </ChainItem>
        </CertificateChain>
        <Conclusion>
            <Indication>PASSED</Indication>
        </Conclusion>
    </BasicBuildingBlocks>
    <TLAnalysis CountryCode="EU" URL="https://ec.europa.eu/tools/lotl/eu-lotl.xml" Id
="LOTL-EC2AE37FE9A43B48B1CFE2A57EBEE2BD6373EDFF36537EB1BC905747ACBF4C3B" Title="List
Of Trusted Lists EU">
        <Constraint>
            <Name NameId="QUAL_TL_FRESH">Is the trusted list fresh?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_EXP">Is the trusted list not expired?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_VERSION">Does the trusted list have the expected
version?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_WS">Is the trusted list well signed?</Name>
            <Status>OK</Status>
        </Constraint>
        <Conclusion>
            <Indication>PASSED</Indication>
        </Conclusion>
    </TLAnalysis>
```

```
    <TLAnalysis CountryCode="BE" URL="https://tsl.belgium.be/tsl-be.xml" Id="TL-
61C0487109BE27255C19CFF26D8F56BEA621E7F381A7B4CBE7FB4750BD477BF9" Title="Trusted List
BE">
        <Constraint>
            <Name NameId="QUAL_TL_FRESH">Is the trusted list fresh?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_EXP">Is the trusted list not expired?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_VERSION">Does the trusted list have the expected
version?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_WS">Is the trusted list well signed?</Name>
            <Status>OK</Status>
        </Constraint>
        <Conclusion>
            <Indication>PASSED</Indication>
        </Conclusion>
    </TLAnalysis>
</DetailedReport>
```

For example the Basic Building Blocks are divided into seven elements:

- FC - Format Checking
- ISC - Identification of the Signing Certificate
- VCI - Validation Context Initialization
- RFC - Revocation Freshness Checker
- XCV - X.509 certificate validation
- CV - Cryptographic Verification
- SAV - Signature Acceptance Validation

The following additional elements also can be executed in case of validation in the past :

- PCV - Past Certificate Validation
- VTS - Validation Time Sliding process
- POE extraction - Proof Of Existence extraction
- PSV - Past Signature Validation

Past certificate/signature validation is used when basic validation of a certificate/signature fails at the current time with an INDETERMINATE status such that the provided proofs of existence may help to go to a determined status. The process shall initialize the *best-signature-time* either to a time

indication for a related POE provided, or the current time when this parameter has not been used by the algorithm.

- **Best-signature-time** is an internal variable for the algorithm denoting the earliest time when it can be trusted by the SVA (either because proven by some POE present in the signature or passed by the DA and for this reason assumed to be trusted) that a signature has existed. [R08]

Each block contains a number of rules that are executed sequentially. The rules are driven by the constraints defined in the validation policy. The result of each rule is OK or NOT OK. The process is stopped when the first rule fails. Each block also contains a conclusion. If all rules are met then the conclusion node indicates PASSED. Otherwise FAILED or INDETERMINATE indication is returned depending on the ETSI standard definition.

Diagnostic Data

This is a data set constructed from the information contained in the signature itself, but also from information retrieved dynamically as revocation data and information extrapolated as the mathematical validity of a signature. All this information is independent of the applied validation policy. Two different validation policies applied to the same diagnostic data can lead to different results.

This is an example of the diagnostic data for a XAdES signature. Certain fields and certain values were trimmed or deleted to make reading easier:

Diagnostic Data

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DiagnosticData xmlns="http://dss.esig.europa.eu/validation/diagnostic">
    <DocumentName>EmptyPage-signed-pades-baseline-b.pdf</DocumentName>
    <ValidationDate>2020-01-21T06:07:03</ValidationDate>
    <Signatures>
        <Signature Id="S-
3723FC8ECE93FD281E21E7239EFAFA0E286306CB5F57F777F5E3A0A3426CA6B1">
            <SignatureFilename>EmptyPage-signed-pades-baseline-
b.pdf</SignatureFilename>
            <ClaimedSigningTime>2019-08-27T14:06:11</ClaimedSigningTime>
            <SignatureFormat>PAdES-BASELINE-B</SignatureFormat>
            <StructuralValidation>
                <Valid>true</Valid>
            </StructuralValidation>
            <DigestMatchers>
                <DigestMatcher type="MESSAGE_DIGEST">
                    <DigestMethod>SHA256</DigestMethod>
                    <DigestValue>
SGEPVFOj/zskv8+nlzixt+PbLxWE9SS67rkpd0V5Wi4=</DigestValue>
                    <DataFound>true</DataFound>
                    <DataIntact>true</DataIntact>
                </DigestMatcher>
            </DigestMatchers>
            <BasicSignature>
```

```
                <EncryptionAlgoUsedToSignThisToken>
RSA</EncryptionAlgoUsedToSignThisToken>
                <KeyLengthUsedToSignThisToken>2048</KeyLengthUsedToSignThisToken>
                <DigestAlgoUsedToSignThisToken>SHA256</DigestAlgoUsedToSignThisToken>
                <SignatureIntact>true</SignatureIntact>
                <SignatureValid>true</SignatureValid>
            </BasicSignature>
            <SigningCertificate Certificate="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F">
                <AttributePresent>true</AttributePresent>
                <DigestValuePresent>true</DigestValuePresent>
                <DigestValueMatch>true</DigestValueMatch>
                <IssuerSerialMatch>true</IssuerSerialMatch>
            </SigningCertificate>
            <CertificateChain>
                <ChainItem Certificate="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F"/>
                <ChainItem Certificate="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA"/>
                <ChainItem Certificate="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
            </CertificateChain>
            <ContentType>1.2.840.113549.1.7.1</ContentType>
            <CommitmentTypeIndication/>
            <Policy>
                <Id>IMPLICIT_POLICY</Id>
                <ZeroHash>false</ZeroHash>
                <Asn1Processable>false</Asn1Processable>
                <Identified>false</Identified>
                <Status>false</Status>
                <DigestAlgorithmsEqual>false</DigestAlgorithmsEqual>
            </Policy>
            <PDFRevision>
                <SignatureFieldName>Signature1</SignatureFieldName>
                <SignerInformationStore>
                    <SignerInfo Processed="true">
                        <Issuer>C=BE,CN=Citizen CA,SERIALNUMBER=201631</Issuer>
                        <SerialNumber>
212676479325593460004449038464688276 73</SerialNumber>
                    </SignerInfo>
                </SignerInformationStore>
                <PDFSignatureDictionary>
                    <SignerName>Pierrick Vandenbroucke (Signature)
70a3cb70f0f4d6513fb12cf0691965c58c7e7679</SignerName>
                    <Type>Sig</Type>
                    <Filter>Adobe.PPKLite</Filter>
                    <SubFilter>ETSI.CAdES.detached</SubFilter>
                    <SignatureByteRange>0 5340 43230 342</SignatureByteRange>
                </PDFSignatureDictionary>
            </PDFRevision>
            <SignerDocumentRepresentations HashOnly="false" DocHashOnly="false"/>
```

```
            <FoundCertificates>
                <RelatedCertificate Certificate="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
                    <Origin>KEY_INFO</Origin>
                </RelatedCertificate>
                <RelatedCertificate Certificate="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F">
                    <Origin>KEY_INFO</Origin>
                    <CertificateRef>
                        <Origin>SIGNING_CERTIFICATE</Origin>

<IssuerSerial>MEswN6Q1MDMxCzAJBgNVBAYTAkJFMRMwEQYDVQQDEwpDaXRpemVuIENBMQ8wDQYDVQQFEwYy
MDE2MzECEBAAAAAAAJKLPMkehMM6uhk=</IssuerSerial>
                        <DigestAlgoAndValue>
                            <DigestMethod>SHA256</DigestMethod>
                            <DigestValue>
T6spAncn5Y5FGO0LauVU0FXwWz2Rl+DRayACjSJ9Gp8=</DigestValue>
                        </DigestAlgoAndValue>
                    </CertificateRef>
                </RelatedCertificate>
                <RelatedCertificate Certificate="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA">
                    <Origin>KEY_INFO</Origin>
                </RelatedCertificate>
            </FoundCertificates>
            <FoundRevocations/>
            <FoundTimestamps/>
            <SignatureScopes>
                <SignatureScope SignerData="D-
BD598965790CA5673910D64EFEDA798485364C66B6F4E7C49D23E1FF8BAFBBE8">
                    <Scope>FULL</Scope>
                    <Name>Full PDF</Name>
                    <Description>Full document</Description>
                </SignatureScope>
            </SignatureScopes>
            <SignatureDigestReference>
                <DigestMethod>SHA256</DigestMethod>
                <DigestValue>
uZKN338SfkpJkk+eM3bEDx3UgdcPW41o7liEALTjb9M=</DigestValue>
            </SignatureDigestReference>

<SignatureValue>m2sMUIIfYKHDVr1IMTyVufWJcwuxwhWjGvkF/xa/rOAKieZCe4LZPa0/uwzgwM4QAbPstd
y4gHSQzCF0R6/fT9hv639kQS3TyZedw1raMeDj9mQOwKOlMl1OIxEI7jSf7xP6n62s0wQAhTJlARnOY1G5vppz
iVKb1vPED27HPBB4Yljn8j6hse+EJ0bwxAN1gwufbxZBvjHYgz/U/9EHafa1oGPcoIBrXvoUdzVX76sVE3nlDv
X4psEU4eq7paIZA7AWGSfWk8/k98pPqFcP2VYJaAju9GI+uZNMfRgPdOvGPxTjUBYiEyr3satod+cMQGiAzie8
0n0ovQrfn7ebcA==</SignatureValue>
        </Signature>
    </Signatures>
    <UsedCertificates>
        <Certificate Id="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA">
```

```
            <SubjectDistinguishedName Format="CANONICAL"
>2.5.4.5=#1306323031363331,cn=citizen ca,c=be</SubjectDistinguishedName>
            <SubjectDistinguishedName Format="RFC2253"
>2.5.4.5=#1306323031363331,CN=Citizen CA,C=BE</SubjectDistinguishedName>
            <IssuerDistinguishedName Format="CANONICAL">cn=belgium root
ca4,c=be</IssuerDistinguishedName>
            <IssuerDistinguishedName Format="RFC2253">CN=Belgium Root
CA4,C=BE</IssuerDistinguishedName>
            <SerialNumber>311326114055543504637452593660348150 60</SerialNumber>
            <CommonName>Citizen CA</CommonName>
            <CountryName>BE</CountryName>
            <AuthorityInformationAccessUrls/>
            <CRLDistributionPoints>
                <crlUrl>http://crl.eid.belgium.be/belgium4.crl</crlUrl>
            </CRLDistributionPoints>
            <OCSPAccessUrls/>
            <Sources>
                <Source>SIGNATURE</Source>
            </Sources>
            <NotAfter>2027-07-25T10:00:00</NotAfter>
            <NotBefore>2015-11-25T10:00:00</NotBefore>
            <PublicKeySize>4096</PublicKeySize>
            <PublicKeyEncryptionAlgo>RSA</PublicKeyEncryptionAlgo>
            <KeyUsageBits>
                <KeyUsage>keyCertSign</KeyUsage>
                <KeyUsage>crlSign</KeyUsage>
            </KeyUsageBits>
            <ExtendedKeyUsages/>
            <IdPkixOcspNoCheck>false</IdPkixOcspNoCheck>
            <BasicSignature>
                <EncryptionAlgoUsedToSignThisToken>
RSA</EncryptionAlgoUsedToSignThisToken>
                <KeyLengthUsedToSignThisToken>4096</KeyLengthUsedToSignThisToken>
                <DigestAlgoUsedToSignThisToken>SHA256</DigestAlgoUsedToSignThisToken>
                <SignatureIntact>true</SignatureIntact>
                <SignatureValid>true</SignatureValid>
            </BasicSignature>
            <SigningCertificate Certificate="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
            <CertificateChain>
                <ChainItem Certificate="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
            </CertificateChain>
            <Trusted>false</Trusted>
            <SelfSigned>false</SelfSigned>
            <CertificatePolicies>
                <certificatePolicy cpsUrl="http://repository.eid.belgium.be"
>2.16.56.12.1.1.2</certificatePolicy>
            </CertificatePolicies>
            <QCStatementIds/>
            <QCTypes/>
```

```
            <TrustedServiceProviders>
                <TrustedServiceProvider TL="TL-
61C0487109BE27255C19CFF26D8F56BEA621E7F381A7B4CBE7FB4750BD477BF9" LOTL="LOTL-
EC2AE37FE9A43B48B1CFE2A57EBEE2BD6373EDFF36537EB1BC905747ACBF4C3B">
                    <TSPNames>
                        <TSPName lang="en">Certipost n.v./s.a.</TSPName>
                    </TSPNames>
                    <TSPRegistrationIdentifiers>
                        <TSPRegistrationIdentifier>VATBE-
0475396406</TSPRegistrationIdentifier>
                    </TSPRegistrationIdentifiers>
                    <TrustedServices>
                        <TrustedService ServiceDigitalIdentifier="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
                            <ServiceNames>
                                <ServiceName lang="en">CN=Belgium Root CA4,
C=BE</ServiceName>
                            </ServiceNames>
                            <ServiceType>
http://uri.etsi.org/TrstSvc/Svctype/CA/QC</ServiceType>

<Status>http://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/granted</Status>
                            <StartDate>2016-06-30T22:00:00</StartDate>
                            <AdditionalServiceInfoUris>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/RootCA-
QC</AdditionalServiceInfoUri>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/ForeSigna
tures</AdditionalServiceInfoUri>
                            </AdditionalServiceInfoUris>
                        </TrustedService>
                        <TrustedService ServiceDigitalIdentifier="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
                            <ServiceNames>
                                <ServiceName lang="en">CN=Belgium Root CA4,
C=BE</ServiceName>
                            </ServiceNames>
                            <ServiceType>
http://uri.etsi.org/TrstSvc/Svctype/CA/QC</ServiceType>

<Status>http://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/undersupervision</Status>
                            <StartDate>2013-06-26T12:00:00</StartDate>
                            <EndDate>2016-06-30T22:00:00</EndDate>
                            <AdditionalServiceInfoUris>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/RootCA-
QC</AdditionalServiceInfoUri>
                            </AdditionalServiceInfoUris>
                        </TrustedService>
                    </TrustedServices>
```

```
                </TrustedServiceProvider>
            </TrustedServiceProviders>
            <Revocations>
                <CertificateRevocation Revocation="R-
4B614C6CFB8CF6B1F9E9C74E07464CE6483EA31E0C3D58A7D4C56D540EDF40FB">
                    <Status>true</Status>
                </CertificateRevocation>
            </Revocations>
            <DigestAlgoAndValue>
                <DigestMethod>SHA256</DigestMethod>
                <DigestValue>
KT0Lo6MeXYKo4/rhJwmTL/3aREI+D3M/sB7xI+c+tNo=</DigestValue>
            </DigestAlgoAndValue>
        </Certificate>
        <Certificate Id="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F">
            <SubjectDistinguishedName Format="CANONICAL"
>2.5.4.5=#130b383730313237333037333 8,2.5.4.42=#130d506965727269636b205061636f,2.5.4.4=
#130d56616e64656e62726f75636b65,cn=pierrick vandenbroucke
(signature),c=be</SubjectDistinguishedName>
            <SubjectDistinguishedName Format="RFC2253"
>2.5.4.5=#130b383730313237333037333 8,2.5.4.42=#130d506965727269636b205061636f,2.5.4.4=
#130d56616e64656e62726f75636b65,CN=Pierrick Vandenbroucke
(Signature),C=BE</SubjectDistinguishedName>
            <IssuerDistinguishedName Format="CANONICAL"
>2.5.4.5=#1306323031363331,cn=citizen ca,c=be</IssuerDistinguishedName>
            <IssuerDistinguishedName Format="RFC2253"
>2.5.4.5=#1306323031363331,CN=Citizen CA,C=BE</IssuerDistinguishedName>
            <SerialNumber>212676479325593460004449038464688276 73</SerialNumber>
            <CommonName>Pierrick Vandenbroucke (Signature)</CommonName>
            <CountryName>BE</CountryName>
            <GivenName>Pierrick Paco</GivenName>
            <Surname>Vandenbroucke</Surname>
            <AuthorityInformationAccessUrls>
                <aiaUrl>http://certs.eid.belgium.be/belgiumrs4.crt</aiaUrl>
            </AuthorityInformationAccessUrls>
            <CRLDistributionPoints>
                <crlUrl>http://crl.eid.belgium.be/eidc201631.crl</crlUrl>
            </CRLDistributionPoints>
            <OCSPAccessUrls>
                <ocspServerUrl>http://ocsp.eid.belgium.be/2</ocspServerUrl>
            </OCSPAccessUrls>
            <Sources>
                <Source>SIGNATURE</Source>
            </Sources>
            <NotAfter>2027-01-21T23:59:59</NotAfter>
            <NotBefore>2017-01-25T22:12:12</NotBefore>
            <PublicKeySize>2048</PublicKeySize>
            <PublicKeyEncryptionAlgo>RSA</PublicKeyEncryptionAlgo>
            <KeyUsageBits>
                <KeyUsage>nonRepudiation</KeyUsage>
```

```
            </KeyUsageBits>
            <ExtendedKeyUsages/>
            <IdPkixOcspNoCheck>false</IdPkixOcspNoCheck>
            <BasicSignature>
                <EncryptionAlgoUsedToSignThisToken>
RSA</EncryptionAlgoUsedToSignThisToken>
                <KeyLengthUsedToSignThisToken>4096</KeyLengthUsedToSignThisToken>
                <DigestAlgoUsedToSignThisToken>SHA256</DigestAlgoUsedToSignThisToken>
                <SignatureIntact>true</SignatureIntact>
                <SignatureValid>true</SignatureValid>
            </BasicSignature>
            <SigningCertificate Certificate="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA"/>
            <CertificateChain>
                <ChainItem Certificate="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA"/>
                <ChainItem Certificate="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
            </CertificateChain>
            <Trusted>false</Trusted>
            <SelfSigned>false</SelfSigned>
            <CertificatePolicies>
                <certificatePolicy cpsUrl="http://repository.eid.belgium.be"
>2.16.56.12.1.1.2.1</certificatePolicy>
            </CertificatePolicies>
            <QCStatementIds>
                <qcStatementOid Description="qc-compliance">
0.4.0.1862.1.1</qcStatementOid>
                <qcStatementOid Description="qc-sscd">0.4.0.1862.1.4</qcStatementOid>
            </QCStatementIds>
            <QCTypes/>
            <TrustedServiceProviders>
                <TrustedServiceProvider TL="TL-
61C0487109BE27255C19CFF26D8F56BEA621E7F381A7B4CBE7FB4750BD477BF9" LOTL="LOTL-
EC2AE37FE9A43B48B1CFE2A57EBEE2BD6373EDFF36537EB1BC905747ACBF4C3B">
                    <TSPNames>
                        <TSPName lang="en">Certipost n.v./s.a.</TSPName>
                    </TSPNames>
                    <TSPRegistrationIdentifiers>
                        <TSPRegistrationIdentifier>VATBE-
0475396406</TSPRegistrationIdentifier>
                    </TSPRegistrationIdentifiers>
                    <TrustedServices>
                        <TrustedService ServiceDigitalIdentifier="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
                            <ServiceNames>
                                <ServiceName lang="en">CN=Belgium Root CA4,
C=BE</ServiceName>
                            </ServiceNames>
                            <ServiceType>
http://uri.etsi.org/TrstSvc/Svctype/CA/QC</ServiceType>
```

```
<Status>http://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/granted</Status>
                            <StartDate>2016-06-30T22:00:00</StartDate>
                            <CapturedQualifiers>

<Qualifier>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/QCQSCDStatusAsInCert</Qu
alifier>
                            </CapturedQualifiers>
                            <AdditionalServiceInfoUris>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/RootCA-
QC</AdditionalServiceInfoUri>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/ForeSigna
tures</AdditionalServiceInfoUri>
                            </AdditionalServiceInfoUris>
                        </TrustedService>
                    </TrustedServices>
                </TrustedServiceProvider>
            </TrustedServiceProviders>
            <Revocations>
                <CertificateRevocation Revocation="R-
5E2868FF9EE4FC069B79171D768B0A90AB137847ADD4F5344EB5F153BB1F19C9">
                    <Status>true</Status>
                </CertificateRevocation>
            </Revocations>
            <DigestAlgoAndValue>
                <DigestMethod>SHA256</DigestMethod>
                <DigestValue>
T6spAncn5Y5FGO0LauVU0FXwWz2Rl+DRayACjSJ9Gp8=</DigestValue>
            </DigestAlgoAndValue>
        </Certificate>
        <Certificate Id="C-
702DD5C1A093CF0A9D71FADD9BF9A7C5857D89FB73B716E867228B3C2BEB968F">
            <SubjectDistinguishedName Format="CANONICAL">cn=belgium root
ca4,c=be</SubjectDistinguishedName>
            <SubjectDistinguishedName Format="RFC2253">CN=Belgium Root
CA4,C=BE</SubjectDistinguishedName>
            <IssuerDistinguishedName Format="CANONICAL">cn=belgium root
ca4,c=be</IssuerDistinguishedName>
            <IssuerDistinguishedName Format="RFC2253">CN=Belgium Root
CA4,C=BE</IssuerDistinguishedName>
            <SerialNumber>5706940941790920504</SerialNumber>
            <CommonName>Belgium Root CA4</CommonName>
            <CountryName>BE</CountryName>
            <AuthorityInformationAccessUrls/>
            <CRLDistributionPoints/>
            <OCSPAccessUrls/>
            <Sources>
                <Source>SIGNATURE</Source>
                <Source>TRUSTED_LIST</Source>
```

```
            </Sources>
            <NotAfter>2032-10-22T12:00:00</NotAfter>
            <NotBefore>2013-06-26T12:00:00</NotBefore>
            <PublicKeySize>4096</PublicKeySize>
            <PublicKeyEncryptionAlgo>RSA</PublicKeyEncryptionAlgo>
            <KeyUsageBits>
                <KeyUsage>keyCertSign</KeyUsage>
                <KeyUsage>crlSign</KeyUsage>
            </KeyUsageBits>
            <ExtendedKeyUsages/>
            <IdPkixOcspNoCheck>false</IdPkixOcspNoCheck>
            <BasicSignature>
                <EncryptionAlgoUsedToSignThisToken>
RSA</EncryptionAlgoUsedToSignThisToken>
                <KeyLengthUsedToSignThisToken>4096</KeyLengthUsedToSignThisToken>
                <DigestAlgoUsedToSignThisToken>SHA256</DigestAlgoUsedToSignThisToken>
                <SignatureIntact>true</SignatureIntact>
                <SignatureValid>true</SignatureValid>
            </BasicSignature>
            <Trusted>true</Trusted>
            <SelfSigned>true</SelfSigned>
            <CertificatePolicies>
                <certificatePolicy cpsUrl="http://repository.eid.belgium.be"
>2.16.56.12.1.1</certificatePolicy>
            </CertificatePolicies>
            <QCStatementIds/>
            <QCTypes/>
            <TrustedServiceProviders>
                <TrustedServiceProvider TL="TL-
61C0487109BE27255C19CFF26D8F56BEA621E7F381A7B4CBE7FB4750BD477BF9" LOTL="LOTL-
EC2AE37FE9A43B48B1CFE2A57EBEE2BD6373EDFF36537EB1BC905747ACBF4C3B">
                    <TSPNames>
                        <TSPName lang="en">Certipost n.v./s.a.</TSPName>
                    </TSPNames>
                    <TSPRegistrationIdentifiers>
                        <TSPRegistrationIdentifier>VATBE-
0475396406</TSPRegistrationIdentifier>
                    </TSPRegistrationIdentifiers>
                    <TrustedServices>
                        <TrustedService ServiceDigitalIdentifier="C-
702DD5C1A093CF0A9D71FADD9BF9A7C5857D89FB73B716E867228B3C2BEB968F">
                            <ServiceNames>
                                <ServiceName lang="en">CN=Belgium Root CA4,
C=BE</ServiceName>
                            </ServiceNames>
                            <ServiceType>
http://uri.etsi.org/TrstSvc/Svctype/CA/QC</ServiceType>

<Status>http://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/granted</Status>
                            <StartDate>2016-06-30T22:00:00</StartDate>
                            <AdditionalServiceInfoUris>
```

```
<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/RootCA-
QC</AdditionalServiceInfoUri>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/ForeSigna
tures</AdditionalServiceInfoUri>
                            </AdditionalServiceInfoUris>
                        </TrustedService>
                        <TrustedService ServiceDigitalIdentifier="C-
702DD5C1A093CF0A9D71FADD9BF9A7C5857D89FB73B716E867228B3C2BEB968F">
                            <ServiceNames>
                                <ServiceName lang="en">CN=Belgium Root CA4,
C=BE</ServiceName>
                            </ServiceNames>
                            <ServiceType>
http://uri.etsi.org/TrstSvc/Svctype/CA/QC</ServiceType>

<Status>http://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/undersupervision</Status>
                            <StartDate>2013-06-26T12:00:00</StartDate>
                            <EndDate>2016-06-30T22:00:00</EndDate>
                            <AdditionalServiceInfoUris>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/RootCA-
QC</AdditionalServiceInfoUri>
                            </AdditionalServiceInfoUris>
                        </TrustedService>
                    </TrustedServices>
                </TrustedServiceProvider>
            </TrustedServiceProviders>
            <Revocations/>
            <DigestAlgoAndValue>
                <DigestMethod>SHA256</DigestMethod>
                <DigestValue>
cC3VwaCTzwqdcfrdm/mnxYV9iftztxboZyKLPCvrlo8=</DigestValue>
            </DigestAlgoAndValue>
        </Certificate>
        <Certificate Id="C-
B336CAA1F3C4930E4EF9C803C12877A004991EE9206C0D4AD3891688C1E478FF">
            <SubjectDistinguishedName Format="CANONICAL">c=be,cn=belgium ocsp
responder</SubjectDistinguishedName>
            <SubjectDistinguishedName Format="RFC2253">C=BE,CN=Belgium OCSP
Responder</SubjectDistinguishedName>
            <IssuerDistinguishedName Format="CANONICAL"
>2.5.4.5=#1306323031363331,cn=citizen ca,c=be</IssuerDistinguishedName>
            <IssuerDistinguishedName Format="RFC2253"
>2.5.4.5=#1306323031363331,CN=Citizen CA,C=BE</IssuerDistinguishedName>
            <SerialNumber>48357032784600921550098 66</SerialNumber>
            <CommonName>Belgium OCSP Responder</CommonName>
            <CountryName>BE</CountryName>
            <AuthorityInformationAccessUrls/>
            <CRLDistributionPoints/>
```

```
            <OCSPAccessUrls/>
            <Sources>
                <Source>OCSP_RESPONSE</Source>
            </Sources>
            <NotAfter>2021-01-29T11:00:00</NotAfter>
            <NotBefore>2019-12-10T11:00:00</NotBefore>
            <PublicKeySize>2048</PublicKeySize>
            <PublicKeyEncryptionAlgo>RSA</PublicKeyEncryptionAlgo>
            <KeyUsageBits>
                <KeyUsage>digitalSignature</KeyUsage>
            </KeyUsageBits>
            <ExtendedKeyUsages>
                <extendedKeyUsagesOid Description="ocspSigning">
1.3.6.1.5.5.7.3.9</extendedKeyUsagesOid>
            </ExtendedKeyUsages>
            <IdPkixOcspNoCheck>true</IdPkixOcspNoCheck>
            <BasicSignature>
                <EncryptionAlgoUsedToSignThisToken>
RSA</EncryptionAlgoUsedToSignThisToken>
                <KeyLengthUsedToSignThisToken>4096</KeyLengthUsedToSignThisToken>
                <DigestAlgoUsedToSignThisToken>SHA256</DigestAlgoUsedToSignThisToken>
                <SignatureIntact>true</SignatureIntact>
                <SignatureValid>true</SignatureValid>
            </BasicSignature>
            <SigningCertificate Certificate="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA"/>
            <CertificateChain>
                <ChainItem Certificate="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA"/>
                <ChainItem Certificate="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
            </CertificateChain>
            <Trusted>false</Trusted>
            <SelfSigned>false</SelfSigned>
            <CertificatePolicies/>
            <QCStatementIds/>
            <QCTypes/>
            <TrustedServiceProviders>
                <TrustedServiceProvider TL="TL-
61C0487109BE27255C19CFF26D8F56BEA621E7F381A7B4CBE7FB4750BD477BF9" LOTL="LOTL-
EC2AE37FE9A43B48B1CFE2A57EBEE2BD6373EDFF36537EB1BC905747ACBF4C3B">
                    <TSPNames>
                        <TSPName lang="en">Certipost n.v./s.a.</TSPName>
                    </TSPNames>
                    <TSPRegistrationIdentifiers>
                        <TSPRegistrationIdentifier>VATBE-
0475396406</TSPRegistrationIdentifier>
                    </TSPRegistrationIdentifiers>
                    <TrustedServices>
                        <TrustedService ServiceDigitalIdentifier="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
```

```
                            <ServiceNames>
                                <ServiceName lang="en">CN=Belgium Root CA4,
C=BE</ServiceName>
                            </ServiceNames>
                            <ServiceType>
http://uri.etsi.org/TrstSvc/Svctype/CA/QC</ServiceType>

<Status>http://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/granted</Status>
                            <StartDate>2016-06-30T22:00:00</StartDate>
                            <AdditionalServiceInfoUris>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/RootCA-
QC</AdditionalServiceInfoUri>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/ForeSigna
tures</AdditionalServiceInfoUri>
                            </AdditionalServiceInfoUris>
                        </TrustedService>
                    </TrustedServices>
                </TrustedServiceProvider>
            </TrustedServiceProviders>
            <Revocations/>
            <DigestAlgoAndValue>
                <DigestMethod>SHA256</DigestMethod>
                <DigestValue>
szbKofPEkw5O+cgDwSh3oASZHukgbA1K04kWiMHkeP8=</DigestValue>
            </DigestAlgoAndValue>
        </Certificate>
        <Certificate Id="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
            <SubjectDistinguishedName Format="CANONICAL">cn=belgium root
ca4,c=be</SubjectDistinguishedName>
            <SubjectDistinguishedName Format="RFC2253">CN=Belgium Root
CA4,C=BE</SubjectDistinguishedName>
            <IssuerDistinguishedName Format="CANONICAL">cn=belgium root
ca4,c=be</IssuerDistinguishedName>
            <IssuerDistinguishedName Format="RFC2253">CN=Belgium Root
CA4,C=BE</IssuerDistinguishedName>
            <SerialNumber>5706940941790920504</SerialNumber>
            <CommonName>Belgium Root CA4</CommonName>
            <CountryName>BE</CountryName>
            <AuthorityInformationAccessUrls/>
            <CRLDistributionPoints/>
            <OCSPAccessUrls/>
            <Sources>
                <Source>SIGNATURE</Source>
                <Source>TRUSTED_LIST</Source>
            </Sources>
            <NotAfter>2028-01-28T12:00:00</NotAfter>
            <NotBefore>2013-06-26T12:00:00</NotBefore>
            <PublicKeySize>4096</PublicKeySize>
```

```
            <PublicKeyEncryptionAlgo>RSA</PublicKeyEncryptionAlgo>
            <KeyUsageBits>
                <KeyUsage>keyCertSign</KeyUsage>
                <KeyUsage>crlSign</KeyUsage>
            </KeyUsageBits>
            <ExtendedKeyUsages/>
            <IdPkixOcspNoCheck>false</IdPkixOcspNoCheck>
            <BasicSignature>
                <EncryptionAlgoUsedToSignThisToken>
RSA</EncryptionAlgoUsedToSignThisToken>
                <KeyLengthUsedToSignThisToken>4096</KeyLengthUsedToSignThisToken>
                <DigestAlgoUsedToSignThisToken>SHA256</DigestAlgoUsedToSignThisToken>
                <SignatureIntact>true</SignatureIntact>
                <SignatureValid>true</SignatureValid>
            </BasicSignature>
            <CertificateChain/>
            <Trusted>true</Trusted>
            <SelfSigned>true</SelfSigned>
            <CertificatePolicies>
                <certificatePolicy cpsUrl="http://repository.eid.belgium.be"
>2.16.56.12.1.1</certificatePolicy>
            </CertificatePolicies>
            <QCStatementIds/>
            <QCTypes/>
            <TrustedServiceProviders>
                <TrustedServiceProvider TL="TL-
61C0487109BE27255C19CFF26D8F56BEA621E7F381A7B4CBE7FB4750BD477BF9" LOTL="LOTL-
EC2AE37FE9A43B48B1CFE2A57EBEE2BD6373EDFF36537EB1BC905747ACBF4C3B">
                    <TSPNames>
                        <TSPName lang="en">Certipost n.v./s.a.</TSPName>
                    </TSPNames>
                    <TSPRegistrationIdentifiers>
                        <TSPRegistrationIdentifier>VATBE-
0475396406</TSPRegistrationIdentifier>
                    </TSPRegistrationIdentifiers>
                    <TrustedServices>
                        <TrustedService ServiceDigitalIdentifier="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
                            <ServiceNames>
                                <ServiceName lang="en">CN=Belgium Root CA4,
C=BE</ServiceName>
                            </ServiceNames>
                            <ServiceType>
http://uri.etsi.org/TrstSvc/Svctype/CA/QC</ServiceType>

<Status>http://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/granted</Status>
                            <StartDate>2016-06-30T22:00:00</StartDate>
                            <AdditionalServiceInfoUris>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/RootCA-
QC</AdditionalServiceInfoUri>
```

```
<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/ForeSigna
tures</AdditionalServiceInfoUri>
                            </AdditionalServiceInfoUris>
                        </TrustedService>
                        <TrustedService ServiceDigitalIdentifier="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
                            <ServiceNames>
                                <ServiceName lang="en">CN=Belgium Root CA4,
C=BE</ServiceName>
                            </ServiceNames>
                            <ServiceType>
http://uri.etsi.org/TrstSvc/Svctype/CA/QC</ServiceType>

<Status>http://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/undersupervision</Status>
                            <StartDate>2013-06-26T12:00:00</StartDate>
                            <EndDate>2016-06-30T22:00:00</EndDate>
                            <AdditionalServiceInfoUris>

<AdditionalServiceInfoUri>http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/RootCA-
QC</AdditionalServiceInfoUri>
                            </AdditionalServiceInfoUris>
                        </TrustedService>
                    </TrustedServices>
                </TrustedServiceProvider>
            </TrustedServiceProviders>
            <Revocations/>
            <DigestAlgoAndValue>
                <DigestMethod>SHA256</DigestMethod>
                <DigestValue>
w/vzclmvCVTu6kKC3RxyJqVOcVD3wposSVujTb/gnKA=</DigestValue>
            </DigestAlgoAndValue>
        </Certificate>
    </UsedCertificates>
    <UsedRevocations>
        <Revocation Id="R-
4B614C6CFB8CF6B1F9E9C74E07464CE6483EA31E0C3D58A7D4C56D540EDF40FB">
            <Origin>EXTERNAL</Origin>
            <Type>CRL</Type>
            <SourceAddress>http://crl.eid.belgium.be/belgium4.crl</SourceAddress>
            <ProductionDate>2020-01-01T11:00:00</ProductionDate>
            <ThisUpdate>2020-01-01T11:00:00</ThisUpdate>
            <NextUpdate>2020-07-31T11:00:00</NextUpdate>
            <CertHashExtensionPresent>false</CertHashExtensionPresent>
            <CertHashExtensionMatch>false</CertHashExtensionMatch>
            <BasicSignature>
                <EncryptionAlgoUsedToSignThisToken>
RSA</EncryptionAlgoUsedToSignThisToken>
                <KeyLengthUsedToSignThisToken>4096</KeyLengthUsedToSignThisToken>
                <DigestAlgoUsedToSignThisToken>SHA256</DigestAlgoUsedToSignThisToken>
                <SignatureIntact>true</SignatureIntact>
```

```
                <SignatureValid>true</SignatureValid>
            </BasicSignature>
            <SigningCertificate Certificate="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
            <CertificateChain>
                <ChainItem Certificate="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
            </CertificateChain>
            <DigestAlgoAndValue>
                <DigestMethod>SHA256</DigestMethod>
                <DigestValue>
S2FMbPuM9rH56cdOB0ZM5kg+ox4MPVin1MVtVA7fQPs=</DigestValue>
            </DigestAlgoAndValue>
        </Revocation>
        <Revocation Id="R-
5E2868FF9EE4FC069B79171D768B0A90AB137847ADD4F5344EB5F153BB1F19C9">
            <Origin>EXTERNAL</Origin>
            <Type>OCSP</Type>
            <SourceAddress>http://ocsp.eid.belgium.be/2</SourceAddress>
            <ProductionDate>2020-01-21T06:07:03</ProductionDate>
            <ThisUpdate>2020-01-21T06:07:03</ThisUpdate>
            <NextUpdate>2020-01-21T06:08:03</NextUpdate>
            <CertHashExtensionPresent>false</CertHashExtensionPresent>
            <CertHashExtensionMatch>false</CertHashExtensionMatch>
            <BasicSignature>
                <EncryptionAlgoUsedToSignThisToken>
RSA</EncryptionAlgoUsedToSignThisToken>
                <KeyLengthUsedToSignThisToken>2048</KeyLengthUsedToSignThisToken>
                <DigestAlgoUsedToSignThisToken>SHA256</DigestAlgoUsedToSignThisToken>
                <SignatureIntact>true</SignatureIntact>
                <SignatureValid>true</SignatureValid>
            </BasicSignature>
            <SigningCertificate Certificate="C-
B336CAA1F3C4930E4EF9C803C12877A004991EE9206C0D4AD3891688C1E478FF"/>
            <CertificateChain>
                <ChainItem Certificate="C-
B336CAA1F3C4930E4EF9C803C12877A004991EE9206C0D4AD3891688C1E478FF"/>
                <ChainItem Certificate="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA"/>
                <ChainItem Certificate="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
            </CertificateChain>
            <DigestAlgoAndValue>
                <DigestMethod>SHA256</DigestMethod>
                <DigestValue>
Xiho/57k/AabeRcddosKkKsTeEet1PU0TrXxU7sfGck=</DigestValue>
            </DigestAlgoAndValue>
        </Revocation>
    </UsedRevocations>
    <UsedTimestamps/>
    <OrphanTokens/>
```

```
    <OriginalDocuments>
        <SignerData Id="D-
BD598965790CA5673910D64EFEDA798485364C66B6F4E7C49D23E1FF8BAFBBE8">
            <ReferencedName>Full PDF</ReferencedName>
            <DigestAlgoAndValue>
                <DigestMethod>SHA256</DigestMethod>
                <DigestValue>
XC6PrKORnL59qX5UJCibBUw/kPLQNtqQx81H+vdlPXw=</DigestValue>
            </DigestAlgoAndValue>
        </SignerData>
    </OriginalDocuments>
    <TrustedLists>
        <TrustedList Id="TL-
61C0487109BE27255C19CFF26D8F56BEA621E7F381A7B4CBE7FB4750BD477BF9">
            <CountryCode>BE</CountryCode>
            <Url>https://tsl.belgium.be/tsl-be.xml</Url>
            <SequenceNumber>45</SequenceNumber>
            <Version>5</Version>
            <LastLoading>2020-01-21T06:03:59</LastLoading>
            <IssueDate>2019-12-17T00:00:00</IssueDate>
            <NextUpdate>2020-06-16T00:00:00</NextUpdate>
            <WellSigned>true</WellSigned>
        </TrustedList>
        <TrustedList Id="LOTL-
EC2AE37FE9A43B48B1CFE2A57EBEE2BD6373EDFF36537EB1BC905747ACBF4C3B" LOTL="true">
            <CountryCode>EU</CountryCode>
            <Url>https://ec.europa.eu/tools/lotl/eu-lotl.xml</Url>
            <SequenceNumber>255</SequenceNumber>
            <Version>5</Version>
            <LastLoading>2020-01-21T06:03:59</LastLoading>
            <IssueDate>2020-01-07T10:00:00</IssueDate>
            <NextUpdate>2020-07-07T00:00:00</NextUpdate>
            <WellSigned>true</WellSigned>
        </TrustedList>
    </TrustedLists>
</DiagnosticData>
```

ETSI Validation Report

The ETSI Validation Report represents an implementation of TS 119 102-2 (cf. [R12]). The report contains a standardized result of an ASiC digital signature validation. It includes the original validation input data, the applied validation policy, as well as the validation result of one or more signature(s) and its(their) constraints.

This is an example of the ETSI validation report:

ETSI Validation Report (TS 119 102-2)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ValidationReport xmlns="http://www.w3.org/2000/09/xmldsig#" xmlns:ns2=
```

```
"http://uri.etsi.org/19102/v1.2.1#" xmlns:ns4="http://uri.etsi.org/02231/v2#"
xmlns:ns3="http://uri.etsi.org/01903/v1.3.2#">
    <ns2:SignatureValidationReport>
        <ns2:SignatureIdentifier id="S-
3723FC8ECE93FD281E21E7239EFAFA0E286306CB5F57F777F5E3A0A3426CA6B1">
            <ns2:DigestAlgAndValue>
                <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                <DigestValue>
SGEPVFOj/zskv8+nlzixt+PbLxWE9SS67rkpd0V5Wi4=</DigestValue>
            </ns2:DigestAlgAndValue>

<SignatureValue>m2sMUIIfYKHDVr1IMTyVufWJcwuxwhWjGvkF/xa/rOAKieZCe4LZPa0/uwzgwM4QAbPstd
y4gHSQzCF0R6/fT9hv639kQS3TyZedw1raMeDj9mQOwKOlMl1OIxEI7jSf7xP6n62s0wQAhTJlARnOY1G5vppz
iVKb1vPED27HPBB4Yljn8j6hse+EJ0bwxAN1gwufbxZBvjHYgz/U/9EHafa1oGPcoIBrXvoUdzVX76sVE3nlDv
X4psEU4eq7paIZA7AWGSfWk8/k98pPqFcP2VYJaAju9GI+uZNMfRgPdOvGPxTjUBYiEyr3satod+cMQGiAzie8
0n0ovQrfn7ebcA==</SignatureValue>
            <ns2:HashOnly>false</ns2:HashOnly>
            <ns2:DocHashOnly>false</ns2:DocHashOnly>
        </ns2:SignatureIdentifier>
        <ns2:ValidationConstraintsEvaluationReport>
            <ns2:ValidationConstraint>
                <ns2:ValidationConstraintIdentifier>
urn:cef:dss:bbb:formatChecking</ns2:ValidationConstraintIdentifier>
                <ns2:ConstraintStatus>
                    <ns2:Status>urn:etsi:019102:constraintStatus:applied</ns2:Status>
                </ns2:ConstraintStatus>
                <ns2:ValidationStatus>
                    <ns2:MainIndication>
urn:etsi:019102:mainindication:passed</ns2:MainIndication>
                </ns2:ValidationStatus>
            </ns2:ValidationConstraint>
            ...
            <ns2:ValidationConstraint>

<ns2:ValidationConstraintIdentifier>urn:cef:dss:bbb:validationTimeSliding</ns2:Validat
ionConstraintIdentifier>
                <ns2:ConstraintStatus>
                    <ns2:Status>urn:etsi:019102:constraintStatus:disabled</ns2:Status>
                </ns2:ConstraintStatus>
            </ns2:ValidationConstraint>
        </ns2:ValidationConstraintsEvaluationReport>
        <ns2:ValidationTimeInfo>
            <ns2:ValidationTime>2020-01-21T06:07:03Z</ns2:ValidationTime>
            <ns2:BestSignatureTime>
                <ns2:POETime>2020-01-21T06:07:03Z</ns2:POETime>
                <ns2:TypeOfProof>urn:etsi:019102:poetype:validation</ns2:TypeOfProof>
            </ns2:BestSignatureTime>
        </ns2:ValidationTimeInfo>
        <ns2:SignersDocument>
            <ns2:DigestAlgAndValue>
                <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
```

```
                <DigestValue>
XC6PrKORnL59qX5UJCibBUw/kPLQNtqQx81H+vdlPXw=</DigestValue>
            </ns2:DigestAlgAndValue>
            <ns2:SignersDocumentRef VOReference="D-
BD598965790CA5673910D64EFEDA798485364C66B6F4E7C49D23E1FF8BAFBBE8"/>
        </ns2:SignersDocument>
        <ns2:SignatureAttributes>
            <ns2:SigningTime Signed="true">
                <ns2:Time>2019-08-27T14:06:11Z</ns2:Time>
            </ns2:SigningTime>
            <ns2:SigningCertificate Signed="true">
                <ns2:AttributeObject VOReference="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F"/>
                <ns2:CertID>

<ns2:X509IssuerSerial>MEswN6Q1MDMxCzAJBgNVBAYTAkJFMRMwEQYDVQQDEwpDaXRpemVuIENBMQ8wDQYD
VQQFEwYyMDE2MzECEBAAAAAAAJKLPMkehMM6uhk=</ns2:X509IssuerSerial>
                    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256
"/>
                    <DigestValue>
T6spAncn5Y5FGO0LauVU0FXwWz2Rl+DRayACjSJ9Gp8=</DigestValue>
                </ns2:CertID>
            </ns2:SigningCertificate>
            <ns2:DataObjectFormat Signed="true">
                <ns2:ContentType>1.2.840.113549.1.7.1</ns2:ContentType>
            </ns2:DataObjectFormat>
            <ns2:MessageDigest Signed="true">
                <ns2:Digest>SGEPVFOj/zskv8+nlzixt+PbLxWE9SS67rkpd0V5Wi4=</ns2:Digest>
            </ns2:MessageDigest>
            <ns2:Name>
                <ns2:NameElement>Pierrick Vandenbroucke (Signature)
70a3cb70f0f4d6513fb12cf0691965c58c7e7679</ns2:NameElement>
            </ns2:Name>
            <ns2:SubFilter>
                <ns2:SubFilterElement>ETSI.CAdES.detached</ns2:SubFilterElement>
            </ns2:SubFilter>
            <ns2:ByteRange>0 5340 43230 342</ns2:ByteRange>
            <ns2:Filter>
                <ns2:Filter>Adobe.PPKLite</ns2:Filter>
            </ns2:Filter>
        </ns2:SignatureAttributes>
        <ns2:SignerInformation Pseudonym="false">
            <ns2:SignerCertificate VOReference="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F"/>
            <ns2:Signer>Pierrick Vandenbroucke (Signature)</ns2:Signer>
        </ns2:SignerInformation>
        <ns2:SignatureQuality>
            <ns2:SignatureQualityInformation>
urn:cef:dss:signatureQualification:QESig</ns2:SignatureQualityInformation>
        </ns2:SignatureQuality>
        <ns2:SignatureValidationProcess>
```

```
            <ns2:SignatureValidationProcessID>
urn:etsi:019102:validationprocess:LTA</ns2:SignatureValidationProcessID>
        </ns2:SignatureValidationProcess>
        <ns2:SignatureValidationStatus>
            <ns2:MainIndication>urn:etsi:019102:mainindication:total-
passed</ns2:MainIndication>
            <ns2:AssociatedValidationReportData>
                <ns2:TrustAnchor VOReference="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
                <ns2:CertificateChain>
                    <ns2:SigningCertificate VOReference="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F"/>
                    <ns2:IntermediateCertificate VOReference="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA"/>
                    <ns2:TrustAnchor VOReference="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
                </ns2:CertificateChain>
                <ns2:CryptoInformation>
                    <ns2:ValidationObjectId VOReference="S-
3723FC8ECE93FD281E21E7239EFAFA0E286306CB5F57F777F5E3A0A3426CA6B1"/>
                    <ns2:Algorithm>http://www.w3.org/2001/04/xmldsig-more#rsa-
sha256</ns2:Algorithm>
                    <ns2:SecureAlgorithm>true</ns2:SecureAlgorithm>
                    <ns2:NotAfter>2022-12-31T23:00:00Z</ns2:NotAfter>
                </ns2:CryptoInformation>
            </ns2:AssociatedValidationReportData>
        </ns2:SignatureValidationStatus>
    </ns2:SignatureValidationReport>
    <ns2:SignatureValidationObjects>
        <ns2:ValidationObject id="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA">
            <ns2:ObjectType>
urn:etsi:019102:validationObject:certificate</ns2:ObjectType>
            <ns2:ValidationObjectRepresentation>
                <ns2:DigestAlgAndValue>
                    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256
"/>
                    <DigestValue>
KT0Lo6MeXYKo4/rhJwmTL/3aREI+D3M/sB7xI+c+tNo=</DigestValue>
                </ns2:DigestAlgAndValue>
            </ns2:ValidationObjectRepresentation>
            <ns2:POE>
                <ns2:POETime>2020-01-21T06:07:03Z</ns2:POETime>
                <ns2:TypeOfProof>urn:etsi:019102:poetype:validation</ns2:TypeOfProof>
            </ns2:POE>
        </ns2:ValidationObject>
        <ns2:ValidationObject id="C-
4FAB29027727E58E4518ED0B6AE554D055F05B3D9197E0D16B20028D227D1A9F">
            <ns2:ObjectType>
urn:etsi:019102:validationObject:certificate</ns2:ObjectType>
            <ns2:ValidationObjectRepresentation>
```

```
                <ns2:DigestAlgAndValue>
                    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256
"/>
                    <DigestValue>
T6spAncn5Y5FGO0LauVU0FXwWz2Rl+DRayACjSJ9Gp8=</DigestValue>
                </ns2:DigestAlgAndValue>
            </ns2:ValidationObjectRepresentation>
            <ns2:POE>
                <ns2:POETime>2020-01-21T06:07:03Z</ns2:POETime>
                <ns2:TypeOfProof>urn:etsi:019102:poetype:validation</ns2:TypeOfProof>
            </ns2:POE>
        </ns2:ValidationObject>
        <ns2:ValidationObject id="C-
702DD5C1A093CF0A9D71FADD9BF9A7C5857D89FB73B716E867228B3C2BEB968F">
            <ns2:ObjectType>
urn:etsi:019102:validationObject:certificate</ns2:ObjectType>
            <ns2:ValidationObjectRepresentation>
                <ns2:DigestAlgAndValue>
                    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256
"/>
                    <DigestValue>
cC3VwaCTzwqdcfrdm/mnxYV9iftztxboZyKLPCvrlo8=</DigestValue>
                </ns2:DigestAlgAndValue>
            </ns2:ValidationObjectRepresentation>
            <ns2:POE>
                <ns2:POETime>2020-01-21T06:07:03Z</ns2:POETime>
                <ns2:TypeOfProof>urn:etsi:019102:poetype:validation</ns2:TypeOfProof>
            </ns2:POE>
        </ns2:ValidationObject>
        <ns2:ValidationObject id="C-
B336CAA1F3C4930E4EF9C803C12877A004991EE9206C0D4AD3891688C1E478FF">
            <ns2:ObjectType>
urn:etsi:019102:validationObject:certificate</ns2:ObjectType>
            <ns2:ValidationObjectRepresentation>
                <ns2:DigestAlgAndValue>
                    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256
"/>
                    <DigestValue>
szbKofPEkw5O+cgDwSh3oASZHukgbA1K04kWiMHkeP8=</DigestValue>
                </ns2:DigestAlgAndValue>
            </ns2:ValidationObjectRepresentation>
            <ns2:POE>
                <ns2:POETime>2020-01-21T06:07:03Z</ns2:POETime>
                <ns2:TypeOfProof>urn:etsi:019102:poetype:validation</ns2:TypeOfProof>
            </ns2:POE>
        </ns2:ValidationObject>
        <ns2:ValidationObject id="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0">
            <ns2:ObjectType>
urn:etsi:019102:validationObject:certificate</ns2:ObjectType>
            <ns2:ValidationObjectRepresentation>
```

```
                <ns2:DigestAlgAndValue>
                    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256
"/>
                    <DigestValue>
w/vzclmvCVTu6kKC3RxyJqVOcVD3wposSVujTb/gnKA=</DigestValue>
                </ns2:DigestAlgAndValue>
            </ns2:ValidationObjectRepresentation>
            <ns2:POE>
                <ns2:POETime>2020-01-21T06:07:03Z</ns2:POETime>
                <ns2:TypeOfProof>urn:etsi:019102:poetype:validation</ns2:TypeOfProof>
            </ns2:POE>
        </ns2:ValidationObject>
        <ns2:ValidationObject id="R-
4B614C6CFB8CF6B1F9E9C74E07464CE6483EA31E0C3D58A7D4C56D540EDF40FB">
            <ns2:ObjectType>urn:etsi:019102:validationObject:CRL</ns2:ObjectType>
            <ns2:ValidationObjectRepresentation>
                <ns2:DigestAlgAndValue>
                    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256
"/>
                    <DigestValue>
S2FMbPuM9rH56cdOB0ZM5kg+ox4MPVin1MVtVA7fQPs=</DigestValue>
                </ns2:DigestAlgAndValue>
            </ns2:ValidationObjectRepresentation>
            <ns2:POE>
                <ns2:POETime>2020-01-21T06:07:03Z</ns2:POETime>
                <ns2:TypeOfProof>urn:etsi:019102:poetype:validation</ns2:TypeOfProof>
            </ns2:POE>
            <ns2:ValidationReport>
                <ns2:ValidationConstraintsEvaluationReport>
                    <ns2:ValidationConstraint>

<ns2:ValidationConstraintIdentifier>urn:cef:dss:bbb:formatChecking</ns2:ValidationCons
traintIdentifier>
                        <ns2:ConstraintStatus>
                            <ns2:Status>
urn:etsi:019102:constraintStatus:disabled</ns2:Status>
                        </ns2:ConstraintStatus>
                    </ns2:ValidationConstraint>
                    ...
                    <ns2:ValidationConstraint>

<ns2:ValidationConstraintIdentifier>urn:cef:dss:bbb:x509CertificateValidation</ns2:Val
idationConstraintIdentifier>
                        <ns2:ConstraintStatus>
                            <ns2:Status>
urn:etsi:019102:constraintStatus:applied</ns2:Status>
                        </ns2:ConstraintStatus>
                        <ns2:ValidationStatus>
                            <ns2:MainIndication>
urn:etsi:019102:mainindication:passed</ns2:MainIndication>
                        </ns2:ValidationStatus>
```

```
                    </ns2:ValidationConstraint>
                </ns2:ValidationConstraintsEvaluationReport>
                <ns2:SignerInformation>
                    <ns2:SignerCertificate VOReference="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
                    <ns2:Signer>Belgium Root CA4</ns2:Signer>
                </ns2:SignerInformation>
                <ns2:SignatureValidationStatus>
                    <ns2:MainIndication>
urn:etsi:019102:mainindication:passed</ns2:MainIndication>
                    <ns2:AssociatedValidationReportData>
                        <ns2:TrustAnchor VOReference="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
                        <ns2:CertificateChain>
                            <ns2:SigningCertificate VOReference="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
                            <ns2:TrustAnchor VOReference="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
                        </ns2:CertificateChain>
                        <ns2:CryptoInformation>
                            <ns2:ValidationObjectId VOReference="R-
4B614C6CFB8CF6B1F9E9C74E07464CE6483EA31E0C3D58A7D4C56D540EDF40FB"/>
                            <ns2:Algorithm>http://www.w3.org/2001/04/xmldsig-more#rsa-
sha256</ns2:Algorithm>
                            <ns2:SecureAlgorithm>true</ns2:SecureAlgorithm>
                            <ns2:NotAfter>2025-12-31T23:00:00Z</ns2:NotAfter>
                        </ns2:CryptoInformation>
                    </ns2:AssociatedValidationReportData>
                </ns2:SignatureValidationStatus>
            </ns2:ValidationReport>
        </ns2:ValidationObject>
        <ns2:ValidationObject id="R-
5E2868FF9EE4FC069B79171D768B0A90AB137847ADD4F5344EB5F153BB1F19C9">
            <ns2:ObjectType>
urn:etsi:019102:validationObject:OCSPResponse</ns2:ObjectType>
            <ns2:ValidationObjectRepresentation>
                <ns2:DigestAlgAndValue>
                    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256
"/>
                    <DigestValue>
Xiho/57k/AabeRcddosKkKsTeEet1PU0TrXxU7sfGck=</DigestValue>
                </ns2:DigestAlgAndValue>
            </ns2:ValidationObjectRepresentation>
            <ns2:POE>
                <ns2:POETime>2020-01-21T06:07:03Z</ns2:POETime>
                <ns2:TypeOfProof>urn:etsi:019102:poetype:validation</ns2:TypeOfProof>
            </ns2:POE>
            <ns2:ValidationReport>
                <ns2:ValidationConstraintsEvaluationReport>
                    <ns2:ValidationConstraint>
```

```
<ns2:ValidationConstraintIdentifier>urn:cef:dss:bbb:formatChecking</ns2:ValidationCons
traintIdentifier>
                        <ns2:ConstraintStatus>
                            <ns2:Status>
urn:etsi:019102:constraintStatus:disabled</ns2:Status>
                        </ns2:ConstraintStatus>
                    </ns2:ValidationConstraint>
                    ...
                    <ns2:ValidationConstraint>

<ns2:ValidationConstraintIdentifier>urn:cef:dss:bbb:x509CertificateValidation</ns2:Val
idationConstraintIdentifier>
                        <ns2:ConstraintStatus>
                            <ns2:Status>
urn:etsi:019102:constraintStatus:applied</ns2:Status>
                        </ns2:ConstraintStatus>
                        <ns2:ValidationStatus>
                            <ns2:MainIndication>
urn:etsi:019102:mainindication:passed</ns2:MainIndication>
                        </ns2:ValidationStatus>
                    </ns2:ValidationConstraint>
                </ns2:ValidationConstraintsEvaluationReport>
                <ns2:SignerInformation>
                    <ns2:SignerCertificate VOReference="C-
B336CAA1F3C4930E4EF9C803C12877A004991EE9206C0D4AD3891688C1E478FF"/>
                    <ns2:Signer>Belgium OCSP Responder</ns2:Signer>
                </ns2:SignerInformation>
                <ns2:SignatureValidationStatus>
                    <ns2:MainIndication>
urn:etsi:019102:mainindication:passed</ns2:MainIndication>
                    <ns2:AssociatedValidationReportData>
                        <ns2:TrustAnchor VOReference="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
                        <ns2:CertificateChain>
                            <ns2:SigningCertificate VOReference="C-
B336CAA1F3C4930E4EF9C803C12877A004991EE9206C0D4AD3891688C1E478FF"/>
                            <ns2:IntermediateCertificate VOReference="C-
293D0BA3A31E5D82A8E3FAE12709932FFDDA44423E0F733FB01EF123E73EB4DA"/>
                            <ns2:TrustAnchor VOReference="C-
C3FBF37259AF0954EEEA4282DD1C7226A54E7150F7C29A2C495BA34DBFE09CA0"/>
                        </ns2:CertificateChain>
                        <ns2:CryptoInformation>
                            <ns2:ValidationObjectId VOReference="R-
5E2868FF9EE4FC069B79171D768B0A90AB137847ADD4F5344EB5F153BB1F19C9"/>
                            <ns2:Algorithm>http://www.w3.org/2001/04/xmldsig-more#rsa-
sha256</ns2:Algorithm>
                            <ns2:SecureAlgorithm>true</ns2:SecureAlgorithm>
                            <ns2:NotAfter>2022-12-31T23:00:00Z</ns2:NotAfter>
                        </ns2:CryptoInformation>
                    </ns2:AssociatedValidationReportData>
                </ns2:SignatureValidationStatus>
```

```
            </ns2:ValidationReport>
        </ns2:ValidationObject>
        <ns2:ValidationObject id="D-
BD598965790CA5673910D64EFEDA798485364C66B6F4E7C49D23E1FF8BAFBBE8">
            <ns2:ObjectType>
urn:etsi:019102:validationObject:signedData</ns2:ObjectType>
            <ns2:ValidationObjectRepresentation>
                <ns2:DigestAlgAndValue>
                    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256
"/>
                    <DigestValue>
XC6PrKORnL59qX5UJCibBUw/kPLQNtqQx81H+vdlPXw=</DigestValue>
                </ns2:DigestAlgAndValue>
            </ns2:ValidationObjectRepresentation>
            <ns2:POE>
                <ns2:POETime>2020-01-21T06:07:03Z</ns2:POETime>
                <ns2:TypeOfProof>urn:etsi:019102:poetype:validation</ns2:TypeOfProof>
            </ns2:POE>
        </ns2:ValidationObject>
    </ns2:SignatureValidationObjects>
</ns2:ValidationReport>
```

Customised Validation Policy

The validation process may be driven by a set of constraints that are contained in the XML file constraint.xml.

constraint.xml (default policy is provided in validation-policy module)

```
<ConstraintsParameters Name="QES AdESQC TL based" xmlns=
"http://dss.esig.europa.eu/validation/policy">
    <Description>Validate electronic signatures and indicates whether they are
Advanced electronic Signatures (AdES), AdES supported by a Qualified Certificate
(AdES/QC) or a
        Qualified electronic Signature (QES). All certificates and their related
chains supporting the signatures are validated against the EU Member State Trusted
Lists (this includes
        signer's certificate and certificates used to validate certificate validity
status services - CRLs, OCSP, and time-stamps).
    </Description>
    <ContainerConstraints>
        <AcceptableContainerTypes Level="FAIL">
            <Id>ASiC-S</Id>
            <Id>ASiC-E</Id>
        </AcceptableContainerTypes>
<!--        <ZipCommentPresent Level="WARN" /> -->
<!--        <AcceptableZipComment Level="WARN"> -->
<!--            <Id>mimetype=application/vnd.etsi.asic-s+zip</Id> -->
<!--            <Id>mimetype=application/vnd.etsi.asic-e+zip</Id> -->
<!--        </AcceptableZipComment> -->
```

```
        <MimeTypeFilePresent Level="FAIL" />
        <AcceptableMimeTypeFileContent Level="WARN">
            <Id>application/vnd.etsi.asic-s+zip</Id>
            <Id>application/vnd.etsi.asic-e+zip</Id>
        </AcceptableMimeTypeFileContent>
        <ManifestFilePresent Level="FAIL" />
        <AllFilesSigned Level="WARN" />
    </ContainerConstraints>
    <SignatureConstraints>
        <StructuralValidation Level="WARN" />
        <AcceptablePolicies Level="FAIL">
            <Id>ANY_POLICY</Id>
            <Id>NO_POLICY</Id>
        </AcceptablePolicies>
        <PolicyAvailable Level="FAIL" />
        <PolicyHashMatch Level="FAIL" />
        <AcceptableFormats Level="FAIL">
            <Id>*</Id>
        </AcceptableFormats>
        <BasicSignatureConstraints>
            <ReferenceDataExistence Level="FAIL" />
            <ReferenceDataIntact Level="FAIL" />
            <ManifestEntryObjectExistence Level="WARN" />
            <SignatureIntact Level="FAIL" />
            <SignatureDuplicated Level="FAIL" />
            <ProspectiveCertificateChain Level="FAIL" />
            <SignerInformationStore Level="FAIL" />
            <SigningCertificate>
                <Recognition Level="FAIL" />
                <Signature Level="FAIL" />
                <NotExpired Level="FAIL" />
                <AuthorityInfoAccessPresent Level="WARN" />
                <RevocationInfoAccessPresent Level="WARN" />
                <RevocationDataAvailable Level="FAIL" />
                <RevocationDataNextUpdatePresent Level="WARN" />
                <RevocationDataFreshness Level="FAIL" />
                <KeyUsage Level="WARN">
                    <Id>nonRepudiation</Id>
                </KeyUsage>
                <SerialNumberPresent Level="WARN" />
                <NotRevoked Level="FAIL" />
                <NotOnHold Level="FAIL" />
                <NotSelfSigned Level="WARN" />
<!--                <Qualification Level="WARN" /> -->
<!--                <SupportedByQSCD Level="WARN" /> -->
<!--                <IssuedToNaturalPerson Level="INFORM" /> -->
<!--                <IssuedToLegalPerson Level="INFORM" /> -->
                <UsePseudonym Level="INFORM" />
                <Cryptographic />
            </SigningCertificate>
            <CACertificate>
```

```
                <Signature Level="FAIL" />
                <NotExpired Level="FAIL" />
                <RevocationDataAvailable Level="FAIL" />
                <RevocationDataNextUpdatePresent Level="WARN" />
                <RevocationDataFreshness Level="FAIL" />
                <NotRevoked Level="FAIL" />
                <NotOnHold Level="FAIL" />
                <Cryptographic />
            </CACertificate>
            <Cryptographic />
        </BasicSignatureConstraints>
        <SignedAttributes>
            <SigningCertificatePresent Level="FAIL" />
            <CertDigestPresent Level="FAIL" />
            <CertDigestMatch Level="FAIL" />
            <IssuerSerialMatch Level="WARN" />
            <SigningTime Level="FAIL" />
            <MessageDigestOrSignedPropertiesPresent Level="FAIL" />
<!--        <ContentType Level="FAIL" value="1.2.840.113549.1.7.1" />
            <ContentHints Level="FAIL" value="*" />
            <CommitmentTypeIndication Level="FAIL">
                <Id>1.2.840.113549.1.9.16.6.1</Id>
                <Id>1.2.840.113549.1.9.16.6.4</Id>
                <Id>1.2.840.113549.1.9.16.6.5</Id>
                <Id>1.2.840.113549.1.9.16.6.6</Id>
            </CommitmentTypeIndication>
            <SignerLocation Level="FAIL" />
            <ContentTimeStamp Level="FAIL" /> -->
        </SignedAttributes>
        <UnsignedAttributes>
<!--        <CounterSignature Level="IGNORE" /> check presence -->
        </UnsignedAttributes>
    </SignatureConstraints>
    <CounterSignatureConstraints>
        <BasicSignatureConstraints>
            <ReferenceDataExistence Level="FAIL" />
            <ReferenceDataIntact Level="FAIL" />
            <SignatureIntact Level="FAIL" />
            <SignatureDuplicated Level="FAIL" />
            <ProspectiveCertificateChain Level="FAIL" />
            <SigningCertificate>
                <Recognition Level="FAIL" />
                <Signature Level="FAIL" />
                <NotExpired Level="FAIL" />
                <AuthorityInfoAccessPresent Level="WARN" />
                <RevocationInfoAccessPresent Level="WARN" />
                <RevocationDataAvailable Level="FAIL" />
                <RevocationDataNextUpdatePresent Level="WARN" />
                <RevocationDataFreshness Level="FAIL" />
                <KeyUsage Level="WARN">
                    <Id>nonRepudiation</Id>
```

```
                </KeyUsage>
                <SerialNumberPresent Level="WARN" />
                <NotRevoked Level="FAIL" />
                <NotOnHold Level="FAIL" />
                <NotSelfSigned Level="WARN" />
                <UsePseudonym Level="INFORM" />
                <Cryptographic />
            </SigningCertificate>
            <CACertificate>
                <Signature Level="FAIL" />
                <NotExpired Level="FAIL" />
                <RevocationDataAvailable Level="FAIL" />
                <RevocationDataNextUpdatePresent Level="WARN" />
                <RevocationDataFreshness Level="FAIL" />
                <NotRevoked Level="FAIL" />
                <NotOnHold Level="FAIL" />
                <Cryptographic />
            </CACertificate>
            <Cryptographic />
        </BasicSignatureConstraints>
        <SignedAttributes>
            <SigningCertificatePresent Level="FAIL" />
            <CertDigestPresent Level="FAIL" />
            <CertDigestMatch Level="FAIL" />
            <IssuerSerialMatch Level="WARN" />
            <SigningTime Level="FAIL" />
            <MessageDigestOrSignedPropertiesPresent Level="FAIL" />
<!--        <ContentType Level="FAIL" value="1.2.840.113549.1.7.1" />
            <ContentHints Level="FAIL" value="*" />
            <CommitmentTypeIndication Level="FAIL">
                <Id>1.2.840.113549.1.9.16.6.1</Id>
                <Id>1.2.840.113549.1.9.16.6.4</Id>
                <Id>1.2.840.113549.1.9.16.6.5</Id>
                <Id>1.2.840.113549.1.9.16.6.6</Id>
            </CommitmentTypeIndication>
            <SignerLocation Level="FAIL" />
            <ContentTimeStamp Level="FAIL" /> -->
        </SignedAttributes>
    </CounterSignatureConstraints>
    <Timestamp>
        <TimestampDelay Level="IGNORE" Unit="DAYS" Value="0" />
        <RevocationTimeAgainstBestSignatureTime Level="FAIL" />
        <BestSignatureTimeBeforeIssuanceDateOfSigningCertificate Level="FAIL" />
        <Coherence Level="WARN" />
        <BasicSignatureConstraints>
            <ReferenceDataExistence Level="FAIL" />
            <ReferenceDataIntact Level="FAIL" />
            <SignatureIntact Level="FAIL" />
            <ProspectiveCertificateChain Level="FAIL" />
            <SigningCertificate>
                <Recognition Level="FAIL" />
```

```
                <Signature Level="FAIL" />
                <NotExpired Level="FAIL" />
                <RevocationDataAvailable Level="FAIL" />
                <RevocationDataNextUpdatePresent Level="WARN" />
                <RevocationDataFreshness Level="FAIL" />
                <ExtendedKeyUsage Level="WARN">
                    <Id>timeStamping</Id>
                </ExtendedKeyUsage>
                <NotRevoked Level="FAIL" />
                <NotOnHold Level="FAIL" />
                <NotSelfSigned Level="WARN" />
                <Cryptographic />
            </SigningCertificate>
            <CACertificate>
                <Signature Level="FAIL" />
                <NotExpired Level="FAIL" />
                <RevocationDataAvailable Level="WARN" />
                <RevocationDataNextUpdatePresent Level="WARN" />
                <RevocationDataFreshness Level="FAIL" />
                <NotRevoked Level="FAIL" />
                <NotOnHold Level="FAIL" />
                <Cryptographic />
            </CACertificate>
            <Cryptographic />
        </BasicSignatureConstraints>
    </Timestamp>
    <Revocation>
        <RevocationFreshness Level="IGNORE" Unit="DAYS" Value="0" />
        <BasicSignatureConstraints>
            <ReferenceDataExistence Level="FAIL" />
            <ReferenceDataIntact Level="FAIL" />
            <SignatureIntact Level="FAIL" />
            <ProspectiveCertificateChain Level="WARN" />
            <SigningCertificate>
                <Recognition Level="FAIL" />
                <Signature Level="FAIL" />
                <NotExpired Level="FAIL" />
                <RevocationDataAvailable Level="FAIL" />
                <RevocationDataNextUpdatePresent Level="WARN" />
                <RevocationDataFreshness Level="FAIL" />
                <NotRevoked Level="FAIL" />
                <NotOnHold Level="FAIL" />
                <Cryptographic />
            </SigningCertificate>
            <CACertificate>
                <Signature Level="FAIL" />
                <NotExpired Level="FAIL" />
                <RevocationDataAvailable Level="WARN" />
                <RevocationDataNextUpdatePresent Level="WARN" />
                <RevocationDataFreshness Level="FAIL" />
                <NotRevoked Level="FAIL" />
```

```
                <NotOnHold Level="FAIL" />
                <Cryptographic />
            </CACertificate>
            <Cryptographic />
        </BasicSignatureConstraints>
    </Revocation>
    <Cryptographic Level="FAIL">
        <AcceptableEncryptionAlgo>
            <Algo>RSA</Algo>
            <Algo>DSA</Algo>
            <Algo>ECDSA</Algo>
            <Algo>PLAIN-ECDSA</Algo>
        </AcceptableEncryptionAlgo>
        <MiniPublicKeySize>
            <Algo Size="160">DSA</Algo>
            <Algo Size="1024">RSA</Algo>
            <Algo Size="160">ECDSA</Algo>
            <Algo Size="160">PLAIN-ECDSA</Algo>
        </MiniPublicKeySize>
        <AcceptableDigestAlgo>
            <Algo>MD2</Algo>
            <Algo>MD5</Algo>
            <Algo>SHA1</Algo>
            <Algo>SHA224</Algo>
            <Algo>SHA256</Algo>
            <Algo>SHA384</Algo>
            <Algo>SHA512</Algo>
            <Algo>SHA3-224</Algo>
            <Algo>SHA3-256</Algo>
            <Algo>SHA3-384</Algo>
            <Algo>SHA3-512</Algo>
            <Algo>RIPEMD160</Algo>
            <Algo>WHIRLPOOL</Algo>
        </AcceptableDigestAlgo>
        <AlgoExpirationDate Format="yyyy">
            <!-- Digest algorithms -->
            <Algo Date="2005">MD2</Algo> <!-- The same as for MD5 -->
            <Algo Date="2005">MD5</Algo> <!-- ETSI TS 102 176-1 (Historical) V2.1.1
-->
            <Algo Date="2009">SHA1</Algo> <!-- ETSI TS 102 176-1 (Historical) V2.0.0
-->
            <Algo Date="2023">SHA224</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026">SHA256</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026">SHA384</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026">SHA512</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026">SHA3-224</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026">SHA3-256</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026">SHA3-384</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026">SHA3-512</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2011">RIPEMD160</Algo> <!-- ETSI TS 102 176-1 (Historical)
V2.0.0 -->
```

```
            <Algo Date="2015">WHIRLPOOL</Algo> <!-- ETSI 119 312 V1.1.1 -->
            <!-- end Digest algorithms -->
            <!-- Encryption algorithms -->
            <Algo Date="2013" Size="160">DSA</Algo> <!-- ETSI TS 102 176-1
(Historical) V2.1.1 -->
            <Algo Date="2013" Size="192">DSA</Algo> <!-- ETSI TS 102 176-1
(Historical) V2.1.1 -->
            <Algo Date="2023" Size="224">DSA</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026" Size="256">DSA</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2009" Size="1024">RSA</Algo> <!-- ETSI TS 102 176-1
(Historical) V2.0.0 -->
            <Algo Date="2016" Size="1536">RSA</Algo> <!-- ETSI 119 312 V1.1.1 -->
            <Algo Date="2023" Size="1900">RSA</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026" Size="3000">RSA</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2013" Size="160">ECDSA</Algo> <!-- ETSI TS 102 176-1
(Historical) V2.1.1 -->
            <Algo Date="2013" Size="192">ECDSA</Algo> <!-- ETSI TS 102 176-1
(Historical) V2.1.1 -->
            <Algo Date="2016" Size="224">ECDSA</Algo> <!-- ETSI 119 312 V1.1.1 -->
            <Algo Date="2026" Size="256">ECDSA</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026" Size="384">ECDSA</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2026" Size="512">ECDSA</Algo> <!-- ETSI 119 312 V1.3.1 -->
            <Algo Date="2013" Size="160">PLAIN-ECDSA</Algo> <!-- ETSI TS 102 176-1
(Historical) V2.1.1 -->
            <Algo Date="2013" Size="192">PLAIN-ECDSA</Algo> <!-- ETSI TS 102 176-1
(Historical) V2.1.1 -->
            <Algo Date="2016" Size="224">PLAIN-ECDSA</Algo> <!-- ETSI 119 312 V1.1.1
-->
            <Algo Date="2026" Size="256">PLAIN-ECDSA</Algo> <!-- ETSI 119 312 V1.3.1
-->
            <Algo Date="2026" Size="384">PLAIN-ECDSA</Algo> <!-- ETSI 119 312 V1.3.1
-->
            <Algo Date="2026" Size="512">PLAIN-ECDSA</Algo> <!-- ETSI 119 312 V1.3.1
-->
            <!-- end Encryption algorithms -->
        </AlgoExpirationDate>
    </Cryptographic>

    <Model Value="SHELL" />

    <!-- eIDAS REGL 910/EU/2014 -->
    <eIDAS>
        <TLFreshness Level="WARN" Unit="HOURS" Value="6" />
        <TLNotExpired Level="WARN" />
        <TLWellSigned Level="FAIL" />
        <TLVersion Level="FAIL" value="5" />
        <TLConsistency Level="FAIL" />
    </eIDAS>
</ConstraintsParameters>
```

CAdES signature (CMS)

To familiarize yourself with this type of signature it is advisable to read the following document:

- CAdES Specifications (cf. [R02])

To implement this form of signature you can use the XAdES examples. You only need to instantiate the CAdES object service and change the SignatureLevel parameter value. Below is an example of the CAdES-Baseline-B signature:

Signing a file with CAdES

```
// Preparing parameters for the CAdES signature
CAdESSignatureParameters parameters = new CAdESSignatureParameters();
// We choose the level of the signature (-B, -T, -LT, -LTA).
parameters.setSignatureLevel(SignatureLevel.CAdES_BASELINE_B);
// We choose the type of the signature packaging (ENVELOPING, DETACHED).
parameters.setSignaturePackaging(SignaturePackaging.ENVELOPING);
// We set the digest algorithm to use with the signature algorithm. You must use the
// same parameter when you invoke the method sign on the token. The default value is
// SHA256
parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// We set the signing certificate
parameters.setSigningCertificate(privateKey.getCertificate());
// We set the certificate chain
parameters.setCertificateChain(privateKey.getCertificateChain());

// Create common certificate verifier
CommonCertificateVerifier commonCertificateVerifier = new CommonCertificateVerifier();
// Create CAdES xadesService for signature
CAdESService service = new CAdESService(commonCertificateVerifier);

// Get the SignedInfo segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);

// This function obtains the signature value for signed information using the
// private key and specified algorithm
DigestAlgorithm digestAlgorithm = parameters.getDigestAlgorithm();
SignatureValue signatureValue = signingToken.sign(dataToSign, digestAlgorithm,
privateKey);

// We invoke the xadesService to sign the document with the signature value obtained
in
// the previous step.
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

PAdES signature (PDF)

The standard ISO 32000-1 (cf. [R05]) allows defining a file format for portable electronic documents. It is based on PDF 1.7 of Adobe Systems. Concerning the digital signature it supports three operations:

- Adding a digital signature to a document,
- Providing a placeholder field for signatures,
- Checking signatures for validity.

PAdES defines eight different profiles to be used with advanced electronic signature in the meaning of European Union Directive 1999/93/EC (cf. [R06]):

- PAdES Basic - PDF signature as specified in ISO 32000-1 (cf. [R05]). The profile is specified in ETSI EN 319 142 (cf. [R03]).
- PAdES-BES Profile - based upon CAdES-BES as specified in ETSI EN 319 122 (cf. [R02]) with the option of a signature time-stamp (CAdES-T).
- PAdES-EPES profile - based upon CAdES-EPES as specified in ETSI EN 319 122 (cf. [R02]). This profile is the same as the PAdES - BES with the addition of a signature policy identifier and optionally a commitment type indication.
- PAdES-LTV Profile - This profile supports the long term validation of PDF Signatures and can be used in conjunction with the above-mentioned profiles.
- Four other PAdES profiles for XML Content.

To familiarize yourself with this type of signature it is advisable to read the documents referenced above.

Below is an example of code to perform a PAdES-BASELINE-B type signature:

Signing a PDF file with PAdES

```
// Preparing parameters for the PAdES signature
PAdESSignatureParameters parameters = new PAdESSignatureParameters();
// We choose the level of the signature (-B, -T, -LT, -LTA).
parameters.setSignatureLevel(SignatureLevel.PAdES_BASELINE_B);
// We set the digest algorithm to use with the signature algorithm. You must use the
// same parameter when you invoke the method sign on the token. The default value is
// SHA256
parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// We set the signing certificate
parameters.setSigningCertificate(privateKey.getCertificate());
// We set the certificate chain
parameters.setCertificateChain(privateKey.getCertificateChain());

// Create common certificate verifier
CommonCertificateVerifier commonCertificateVerifier = new CommonCertificateVerifier();
// Create PAdESService for signature
PAdESService service = new PAdESService(commonCertificateVerifier);

// Get the SignedInfo segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);

// This function obtains the signature value for signed information using the
// private key and specified algorithm
DigestAlgorithm digestAlgorithm = parameters.getDigestAlgorithm();
SignatureValue signatureValue = signingToken.sign(dataToSign, digestAlgorithm,
privateKey);

// We invoke the xadesService to sign the document with the signature value obtained
in
// the previous step.
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

In order too add a timestamp to the signature (PAdES-T or LTA), a TSP source must be provided to the service.

To create PAdES-BASELINE-B level with additional options: signature policy identifier and optionally a commitment type indication, please observe the following example in code 5.

All these parameters are optional.

Defining a Signature Policy

```
// Instantiate a Policy object
Policy signaturePolicy = new Policy();
// The string representation of the OID of the signature policy to use when signing.
signaturePolicy.setId("1.2.3.4.5.6");
// Defines a description for a signature policy
signaturePolicy.setDescription("Perfect Signature Policy");
// The hash function used to compute the value of the SignaturePolicyHashValue entry.
// Entries must be represented the same way as in table 257 of ISO 32000-1 (cf.
<<R05>>).
signaturePolicy.setDigestAlgorithm(DigestAlgorithm.SHA1);
// The value of the hash of the signature policy, computed the same way as
// in clause 5.2.9 of CAdES (ETSI EN 319 122 (cf. <<R02>>)).
signaturePolicy.setDigestValue(new byte[] { 'd', 'i', 'g', 'e', 's', 't', 'v', 'a', '
l', 'u', 'e' });
// Defines a URI where the policy can be accessed from
signaturePolicy.setSpuri("http://spuri.test");
// Defines a policy qualifier
signaturePolicy.setQualifier("OIDAsURN");
parameters.bLevel().setSignaturePolicy(signaturePolicy);
```

The extension of a signature of the level PAdES-BASELINE-B up to PAdES-BASELINE-LTA profile will add the following features:

- Addition of validation data to an existing PDF document which may be used to validate earlier signatures within the document (including PDF signatures and time-stamp signatures).
- Addition of a document time-stamp which protects the existing document and any validation data.
- Further validation data and document time-stamp may be added to a document over time to maintain its authenticity and integrity.

PAdES Visible Signature

The framework also allows creation of PDF files with visible signature as specified in ETSI EN 319 142 (cf. [R03]). In the `SignatureParameters` object, there's a special attribute named `SignatureImageParameters`. This parameter allows you customize the visual signature (with text, with image or with image and text). Below there is an example of code to perform a PADES-BASELINE-B type signature with a visible signature:

Add a visible signature to a PDF document

```
// Preparing parameters for the PAdES signature
PAdESSignatureParameters parameters = new PAdESSignatureParameters();
// We choose the level of the signature (-B, -T, -LT, -LTA).
parameters.setSignatureLevel(SignatureLevel.PAdES_BASELINE_B);

// We set the signing certificate
parameters.setSigningCertificate(privateKey.getCertificate());
// We set the certificate chain
parameters.setCertificateChain(privateKey.getCertificateChain());

// Initialize visual signature and configure
SignatureImageParameters imageParameters = new SignatureImageParameters();
// set an image
imageParameters.setImage(new InMemoryDocument(getClass().getResourceAsStream(
"/signature-pen.png")));
// the origin is the left and top corner of the page
imageParameters.setxAxis(200);
imageParameters.setyAxis(400);
imageParameters.setWidth(300);
imageParameters.setHeight(200);
parameters.setImageParameters(imageParameters);

// Create common certificate verifier
CommonCertificateVerifier commonCertificateVerifier = new CommonCertificateVerifier();
// Create PAdESService for signature
PAdESService service = new PAdESService(commonCertificateVerifier);
service.setPdfObjFactory(new PdfBoxNativeObjectFactory());
// Get the SignedInfo segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);

// This function obtains the signature value for signed information using the
// private key and specified algorithm
DigestAlgorithm digestAlgorithm = parameters.getDigestAlgorithm();
SignatureValue signatureValue = signingToken.sign(dataToSign, digestAlgorithm,
privateKey);

// We invoke the xadesService to sign the document with the signature value obtained
in
// the previous step.
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

Additionally, DSS also allows you to insert a visible signature to an existing field :

Add a visible signature to an existing field

```
parameters.setSignatureFieldId("field-id");
```

In case of placing an image or text to an existing field, the visible signature will fill out the whole available area of the field.

Visible signature parameters (image and text)

This chapter introduces existing parameters for creation of visible signatures with DSS. DSS has three implementations for visible signature drawing:

- **OpenPDF (iText)** - supports separate image and text drawing;
- **PDFBox Default** - supports separate image and text drawing, as well as a joint drawing of image and text together. Transforms text to an image;
- **PDFBox Native** - supports separate image and text drawing, as well as a joint drawing of image and text together. Prints text in a native way, that increases quality of the produced signature.

Positioning

DSS provides a set of functions allowing to place the signature field on a specific place in the PDF page :

Visible signature positioning

```
// Object containing a list of visible signature parameters
SignatureImageParameters signatureImageParameters = new SignatureImageParameters();

// Allows defining of a specific page in a PDF document where the signature must be
placed.
// The counting of pages starts from 1 (the first page)
// (the default value = 1).
signatureImageParameters.setPage(1);

// Absolute positioning functions, allowing to specify a margin between
// the left page side and the top page side respectively, and
// a signature field (if no rotation and alignment is applied).
signatureImageParameters.setxAxis(10);
signatureImageParameters.setyAxis(10);

// Allows alignment of a signature field horizontally to a page. Allows the following
values:
/* _NONE_ (_DEFAULT value._ None alignment is applied, coordinates are counted from
the left page side);
   _LEFT_ (the signature is aligned to the left side, coordinated are counted from the
left page side);
   _CENTER_ (the signature is aligned to the center of the page, coordinates are
counted automatically);
   _RIGHT_ (the signature is aligned to the right side, coordinated are counted from
the right page side). */
signatureImageParameters.setAlignmentHorizontal(VisualSignatureAlignmentHorizontal.CEN
TER);

// Allows alignment of a signature field vertically to a page. Allows the following
```

```
values:
/* _NONE_ (_DEFAULT value._ None alignment is applied, coordinated are counted from
the top side of a page);
   _TOP_ (the signature is aligned to a top side, coordinated are counted from the top
page side);
   _MIDDLE_ (the signature aligned to a middle of a page, coordinated are counted
automatically);
   _BOTTOM_ (the signature is aligned to a bottom side, coordinated are counted from
the bottom page side). */
signatureImageParameters.setAlignmentVertical(VisualSignatureAlignmentVertical.TOP);

// Rotates the signature field and changes the coordinates' origin respectively to its
values as following:
/* _NONE_ (_DEFAULT value._ No rotation is applied. The origin of coordinates begins
from the top left corner of a page);
   _AUTOMATIC_ (Rotates a signature field respectively to the page's rotation. Rotates
the signature field on the same value as a defined in a PDF page);
   _ROTATE_90_ (Rotates a signature field for a 90&amp;#176; clockwise. Coordinates'
origin begins from top right page corner);
   _ROTATE_180_ (Rotates a signature field for a 180&amp;#176; clockwise. Coordinates'
origin begins from the bottom right page corner);
   _ROTATE_270_ (Rotates a signature field for a 270&amp;#176; clockwise. Coordinates'
origin begins from the bottom left page corner). */
signatureImageParameters.setRotation(VisualSignatureRotation.AUTOMATIC);

padesSignatureParameters.setImageParameters(signatureImageParameters);
```

Dimensions

DSS framework provides a set of functions to manage the signature field size :

Visible signature dimensions

```
// Object containing a list of visible signature parameters
SignatureImageParameters signatureImageParameters = new SignatureImageParameters();

// Allows defining of a specific page in a PDF document where the signature must be
placed.
// The counting of pages starts from 1 (the first page)
// (the default value = 1).
signatureImageParameters.setPage(1);

// Absolute positioning functions, allowing to specify a margin between
// the left page side and the top page side respectively, and
// a signature field (if no rotation and alignment is applied).
signatureImageParameters.setxAxis(10);
signatureImageParameters.setyAxis(10);

// Allows alignment of a signature field horizontally to a page. Allows the following
values:
/* _NONE_ (_DEFAULT value._ None alignment is applied, coordinates are counted from
```

```
the left page side);
   _LEFT_ (the signature is aligned to the left side, coordinated are counted from the
left page side);
   _CENTER_ (the signature is aligned to the center of the page, coordinates are
counted automatically);
   _RIGHT_ (the signature is aligned to the right side, coordinated are counted from
the right page side). */
signatureImageParameters.setAlignmentHorizontal(VisualSignatureAlignmentHorizontal.CEN
TER);

// Allows alignment of a signature field vertically to a page. Allows the following
values:
/* _NONE_ (_DEFAULT value._ None alignment is applied, coordinated are counted from
the top side of a page);
   _TOP_ (the signature is aligned to a top side, coordinated are counted from the top
page side);
   _MIDDLE_ (the signature aligned to a middle of a page, coordinated are counted
automatically);
   _BOTTOM_ (the signature is aligned to a bottom side, coordinated are counted from
the bottom page side). */
signatureImageParameters.setAlignmentVertical(VisualSignatureAlignmentVertical.TOP);

// Rotates the signature field and changes the coordinates' origin respectively to its
values as following:
/* _NONE_ (_DEFAULT value._ No rotation is applied. The origin of coordinates begins
from the top left corner of a page);
   _AUTOMATIC_ (Rotates a signature field respectively to the page's rotation. Rotates
the signature field on the same value as a defined in a PDF page);
   _ROTATE_90_ (Rotates a signature field for a 90&#176; clockwise. Coordinates'
origin begins from top right page corner);
   _ROTATE_180_ (Rotates a signature field for a 180&#176; clockwise. Coordinates'
origin begins from the bottom right page corner);
   _ROTATE_270_ (Rotates a signature field for a 270&#176; clockwise. Coordinates'
origin begins from the bottom left page corner). */
signatureImageParameters.setRotation(VisualSignatureRotation.AUTOMATIC);

padesSignatureParameters.setImageParameters(signatureImageParameters);
```

Text Parameters

The available implementations allow placing of a visible text to a signature field :

List of available visible text parameters

```
// Instantiates a SignatureImageTextParameters object
SignatureImageTextParameters textParameters = new SignatureImageTextParameters();
// Allows you to set a DSSFont object that defines the text style (see more
information in the section "Fonts usage")
textParameters.setFont(font);
// Defines the text content
textParameters.setText("My visual signature \n #1");
// Specifies the text size value (the default font size is 12pt)
textParameters.setSize(14);
// Defines the color of the characters
textParameters.setTextColor(Color.BLUE);
// Defines the background color for the area filled out by the text
textParameters.setBackgroundColor(Color.YELLOW);
// Defines a padding between the text and a border of its bounding area
textParameters.setPadding(20);
// Set textParameters to a SignatureImageParameters object
imageParameters.setTextParameters(textParameters);
```

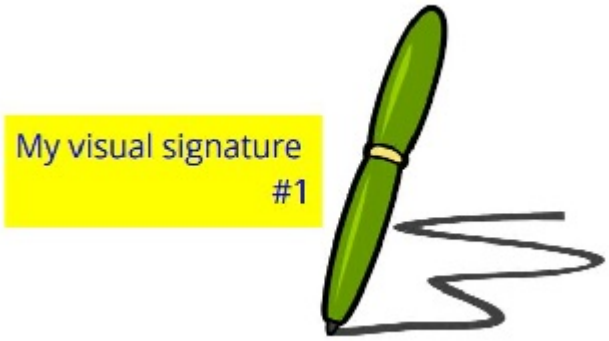
Text and image combination

DSS provides a set of functions to align a text respectively to an image. The parameters must be applied to a 'SignatureImageTextParameters' object :

Combination of text and image parameters

```
// Specifies a text position relatively to an image (Note: applicable only for joint
image+text visible signatures).
// Thus with _SignerPosition.LEFT_ value, the text will be placed on the left side,
// and image will be aligned to the right side inside the signature field
textParameters.setSignerTextPosition(SignerTextPosition.LEFT);
// Specifies a horizontal alignment of a text with respect to its area
textParameters.setSignerTextHorizontalAlignment(SignerTextHorizontalAlignment.RIGHT);
// Specifies a vertical alignment of a text block with respect to a signature field
area
textParameters.setSignerTextVerticalAlignment(SignerTextVerticalAlignment.TOP);
```

The result of applying the foregoing transformations is provided on the image below:



Fonts usage

Since version 5.5, DSS supports two types of fonts. The custom font must be added as an instance of `DSSFont` interface to a `SignatureImageTextParameters` object. `DSSFont` interface has two implementations:

- `DSSFileFont` for using of physical fonts, which must be embedded to the produced PDF document. To create an instance of the class, you must pass to a `DSSFileFont` constructor an object of `DSSDocument` type or InputStream of the font file;
- `DSSJavaFont` for using of logical fonts (default Java fonts). The logical Java fonts allow you to significantly reduce the document size, because these fonts cannot be embedded to the final PDF document. Be aware, because of the fact, using of logical fonts does not allow producing PDF documents satisfying the PDF/A standard. To create an instance of this class, you should pass as an input a `java.awt.Font` object or target font parameters (name, style, size).

You can create a custom font as following (for a physical font):

Add a custom font as a file

```
// Initialize text to generate for visual signature
DSSFileFont font = new DSSFileFont(getClass().getResourceAsStream(
"/fonts/OpenSansRegular.ttf"));
```

or for a logical font:

Java font usage

```
SignatureImageTextParameters textParameters = new SignatureImageTextParameters();
DSSJavaFont font = new DSSJavaFont(new Font(Font.SERIF, Font.PLAIN, 16));
textParameters.setFont(font);
textParameters.setTextColor(Color.BLUE);
textParameters.setText("My visual signature");
imageParameters.setTextParameters(textParameters);
```

By default, DSS uses a Google font : 'PT Serif Regular' (its physical implementation).

'Native PDFBox Drawer' implementation supports only one of the following fonts: SERIF, SANS-SERIF, MONOSPACED, DIALOG and DIALOG_INPUT.

ASiC signature (containers)

When creating a digital signature, the user must choose between different packaging elements, namely enveloping, enveloped or detached. This choice is not obvious, because in one case the signature will alter the signed document and in the other case it is possible to lose the association between the signed document and its signature. That's where the standard ETSI EN 319 162 (cf. [R04]) offers a standardized use of container forms to establish a common way for associating data objects with advanced signatures or time-stamp tokens.

A number of application environments use ZIP based container formats to package sets of files together with meta-information. ASiC technical specification is designed to operate with a range of such ZIP based application environments. Rather than enforcing a single packaging structure, ASiC describes how these package formats can be used to associate advanced electronic signatures with any data objects.

The standard defines two types of containers; the first (ASiC-S) allows you to associate one or more signatures with a single data element. In this case the structure of the signature can be based (in a general way) on a single CAdES signature or on multiple XAdES signatures or finally on a single TST; the second is an extended container (ASiC-E) that includes multiple data objects. Each data object may be signed by one or more signatures which structure is similar to ASiC-S. This second type of container is compatible with OCF, UCF and ODF formats.

For the moment the DSS framework has some restrictions on the containers you can generate, depending on the input file. If the input file is already an ASiC container, the output container must be the same type of container based on the same type of signature. If the input is any other file, the output does not have any restriction.

Table 6. ASiC containers

Input	Output
ASiC-S CAdES	ASiC-S CAdES
ASiC-S XAdES	ASiC-S XAdES
ASiC-E CAdES	ASiC-E CAdES
ASiC-E XAdES	ASiC-E XAdES
Binary	ASiC-S CAdES, ASiC-S XAdES, ASiC-E CAdES, ASiC-E XAdES

This is an example of the source code for signing a document using ASiCS-S based on XAdES-B:

Sign a file within an ASiC-S container

```
// Preparing parameters for the AsicS signature
ASiCWithXAdESSignatureParameters parameters = new ASiCWithXAdESSignatureParameters();
// We choose the level of the signature (-B, -T, -LT, LTA).
parameters.setSignatureLevel(SignatureLevel.XAdES_BASELINE_B);
// We choose the container type (ASiC-S or ASiC-E)
parameters.aSiC().setContainerType(ASiCContainerType.ASiC_S);

// We set the digest algorithm to use with the signature algorithm. You must use the
// same parameter when you invoke the method sign on the token. The default value is
// SHA256
parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// We set the signing certificate
parameters.setSigningCertificate(privateKey.getCertificate());
// We set the certificate chain
parameters.setCertificateChain(privateKey.getCertificateChain());

// Create common certificate verifier
CommonCertificateVerifier commonCertificateVerifier = new CommonCertificateVerifier();
// Create ASiC service for signature
ASiCWithXAdESService service = new ASiCWithXAdESService(commonCertificateVerifier);

// Get the SignedInfo segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(toSignDocument, parameters);

// This function obtains the signature value for signed information using the
// private key and specified algorithm
DigestAlgorithm digestAlgorithm = parameters.getDigestAlgorithm();
SignatureValue signatureValue = signingToken.sign(dataToSign, digestAlgorithm,
privateKey);

// We invoke the xadesService to sign the document with the signature value obtained
in
// the previous step.
DSSDocument signedDocument = service.signDocument(toSignDocument, parameters,
signatureValue);
```

This is another example of the source code for signing multiple documents using ASiCS-E based on CAdES:

Sign multiple files within an ASiC-E container

```
// Preparing the documents to be embedded in the container and signed
List<DSSDocument> documentsToBeSigned = new ArrayList<>();
documentsToBeSigned.add(new FileDocument("src/main/resources/hello-world.pdf"));
documentsToBeSigned.add(new FileDocument("src/main/resources/xml_example.xml"));

// Preparing parameters for the ASiC-E signature
ASiCWithCAdESSignatureParameters parameters = new ASiCWithCAdESSignatureParameters();

// We choose the level of the signature (-B, -T, -LT or -LTA).
parameters.setSignatureLevel(SignatureLevel.CAdES_BASELINE_B);
// We choose the container type (ASiC-S pr ASiC-E)
parameters.aSiC().setContainerType(ASiCContainerType.ASiC_E);

// We set the digest algorithm to use with the signature algorithm. You
// must use the
// same parameter when you invoke the method sign on the token. The
// default value is
// SHA256
parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// We set the signing certificate
parameters.setSigningCertificate(privateKey.getCertificate());
// We set the certificate chain
parameters.setCertificateChain(privateKey.getCertificateChain());

// Create common certificate verifier
CommonCertificateVerifier commonCertificateVerifier = new CommonCertificateVerifier();
// Create ASiC service for signature
ASiCWithCAdESService service = new ASiCWithCAdESService(commonCertificateVerifier);

// Get the SignedInfo segment that need to be signed.
ToBeSigned dataToSign = service.getDataToSign(documentsToBeSigned, parameters);

// This function obtains the signature value for signed information
// using the
// private key and specified algorithm
DigestAlgorithm digestAlgorithm = parameters.getDigestAlgorithm();
SignatureValue signatureValue = signingToken.sign(dataToSign, digestAlgorithm,
privateKey);

// We invoke the xadesService to sign the document with the signature
// value obtained in
// the previous step.
DSSDocument signedDocument = service.signDocument(documentsToBeSigned, parameters,
signatureValue);
```

Please note that you need to pass only few parameters to the service. Other parameters, although are positioned, will be overwritten by the internal implementation of the service. Therefore, the

obtained signature is always based on CAdES and of DETACHED packaging.

It is also possible with the framework DSS to make an extension of an ASiC container to the level XAdES-BASELINE-T or -LT.

Timestamps

Timestamp creation

Since DSS 5.6 the framework allows an independent document timestamping (without a signature). The following Document Signature Services support the timestamping :

- `PAdESService` - adds a timestamp to a PDF document;
- `ASiCWithCAdESService` - creates a timestamped ASiC container with provided documents.

PDF timestamping

```
// Loads a document to be timestamped
DSSDocument documentToTimestamp = new FileDocument(new File("src/main/resources/hello-world.pdf"));

// Configure a PAdES service for PDF timestamping
PAdESService service = new PAdESService(getCompleteCertificateVerifier());
service.setTspSource(getGoodTsa());

// Execute the timestamp method
DSSDocument timestampedDoc = service.timestamp(documentToTimestamp, new PAdESTimestampParameters());
```

Timestamp validation

As well as a single timestamp creation, DSS provides a validation service for timestamped documents. The timestamp validation process represents "5.4 Time-stamp validation building block" (cf. [R08]). The validation process is identical to The signature validation process. An appropriate validator will be selected automatically. In total, DSS supports timestamp-alone validation for the following file formats:

- Detached CMS timestamp (`DetachedTimestampValidator`) - a detached signed content must be provided (or its digest);
- PDF document (`PDFDocumentValidator`);
- ASiC CAdES container with a timestamp (`ASiCEWithCAdESTimestampValidator`).

The validation process can be run with the following inputs :

Timestamped document validation

```
// Load a document validator. The appropriate validator class will be determined
automatically.
SignedDocumentValidator validator = SignedDocumentValidator.fromDocument
(timestampedDoc);
// Configure the validator. Provide a certificate verifier.
validator.setCertificateVerifier(getCompleteCertificateVerifier());
// Validate the document
Reports reports = validator.validateDocument();
```

The produced reports use the same structure as for The signature validation.

Timestamp qualification

DSS is also able to determine a qualification level of a timestamp, if a relative information about TrustServiceProviders is provided to a certificate verifier (loaded automatically to a trusted certificate source with TLValidationJob) (cf. [R13]).

Three qualification levels are supported by DSS and can be obtained :

- QTSA (issued from a granted trust service with TSA/QTST type at the timestamp production time);
- TSA any other from a known trust anchor;
- N/A for others.

An example of a produced Detailed Report you can see below:

Timestamp Detailed Report

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DetailedReport xmlns="http://dss.esig.europa.eu/validation/detailed-report">
    <Timestamp Id="T-1F9F5B92B9DF5678CD57A7B47FEEADDE365017BD67E5E7468A5ADF14F6CC901F
">
        <ValidationProcessTimestamp Type="CONTENT_TIMESTAMP" ProductionTime="2018-05-
23T13:18:29" Title="Validation Process for Timestamps">
            <Constraint Id="T-
1F9F5B92B9DF5678CD57A7B47FEEADDE365017BD67E5E7468A5ADF14F6CC901F">
                <Name NameId="ADEST_ROTVPIIC">Is the result of the timestamps
validation process conclusive?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
        </ValidationProcessTimestamp>
        <ValidationTimestampQualification TimestampQualification="QTSA" Title=
"Timestamp Qualification">
            <Constraint>
                <Name NameId="QUAL_CERT_TRUSTED_LIST_REACHED">Has a trusted list been
reached for the certificate chain?</Name>
```

```
                <Status>OK</Status>
            </Constraint>
            <Constraint Id="LOTL-
EC2AE37FE9A43B48B1CFE2A57EBEE2BD6373EDFF36537EB1BC905747ACBF4C3B">
                <Name NameId="QUAL_TRUSTED_LIST_ACCEPT">Is the trusted list
acceptable?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Trusted List : https://ec.europa.eu/tools/lotl/eu-
lotl.xml</AdditionalInfo>
            </Constraint>
            <Constraint Id="TL-
B66584674C05B72F6BB09FF13C632EB1BA72A0DD0DCF9A6822EF73374F6DFC48">
                <Name NameId="QUAL_TRUSTED_LIST_ACCEPT">Is the trusted list
acceptable?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Trusted List : https://www.nrca-ds.de/st/TSL-
XML.xml</AdditionalInfo>
            </Constraint>
            <Constraint>
                <Name NameId="QUAL_HAS_QTST">Is the certificate related to a
TSA/QTST?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="QUAL_HAS_GRANTED">Is the certificate related to a trust
service with a granted status?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint>
                <Name NameId="QUAL_HAS_GRANTED_AT">Is the certificate related to a
trust service with a granted status at the production time?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
        </ValidationTimestampQualification>
    </Timestamp>
    <BasicBuildingBlocks Id="T-
1F9F5B92B9DF5678CD57A7B47FEEADDE365017BD67E5E7468A5ADF14F6CC901F" Type="TIMESTAMP">
        <ISC Title="Identification of the Signing Certificate">
            <Constraint>
                <Name NameId="BBB_ICS_ISCI">Is there an identified candidate for the
signing certificate?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
            <CertificateChain>
                <ChainItem Id="C-
```

```
9ACF4D4E37AD647B1310D55800234BBF17FFFE01CAF7E63ED7223B770CB0D2E9">
                    <Source>TRUSTED_LIST</Source>
                </ChainItem>
                <ChainItem Id="C-
5F522A439611DEC501F0468FF6A455D23DFBD08EEB33964A0C5EDE95593E1FA8">
                    <Source>TIMESTAMP</Source>
                </ChainItem>
                <ChainItem Id="C-
4D24807B9CAD5110F40ED79D934346D7C9B0290431DC9B11A40BBB86FCF2AEF6">
                    <Source>TIMESTAMP</Source>
                </ChainItem>
            </CertificateChain>
        </ISC>
        <XCV Title="X509 Certificate Validation">
            <Constraint>
                <Name NameId="BBB_XCV_CCCBB">Can the certificate chain be built till a
trust anchor?</Name>
                <Status>OK</Status>
            </Constraint>
            <Constraint Id="C-
9ACF4D4E37AD647B1310D55800234BBF17FFFE01CAF7E63ED7223B770CB0D2E9">
                <Name NameId="BBB_XCV_SUB">Is the certificate validation
conclusive?</Name>
                <Status>OK</Status>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
            <SubXCV Id="C-
9ACF4D4E37AD647B1310D55800234BBF17FFFE01CAF7E63ED7223B770CB0D2E9" TrustAnchor="true"
Title="Certificate">
                <Conclusion>
                    <Indication>PASSED</Indication>
                </Conclusion>
            </SubXCV>
        </XCV>
        <CV Title="Cryptographic Verification">
            <Constraint>
                <Name NameId="BBB_CV_IRDOF">Has the reference data object been
found?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Reference : MESSAGE_IMPRINT</AdditionalInfo>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_CV_IRDOI">Is the reference data object
intact?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Reference : MESSAGE_IMPRINT</AdditionalInfo>
            </Constraint>
            <Constraint>
                <Name NameId="BBB_CV_ISIT">Is timestampss signature intact?</Name>
```

```
                <Status>OK</Status>
                <AdditionalInfo>Id = T-
1F9F5B92B9DF5678CD57A7B47FEEADDE365017BD67E5E7468A5ADF14F6CC901F</AdditionalInfo>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
        </CV>
        <SAV ValidationTime="2020-01-23T13:09:12" Title="Signature Acceptance
Validation">
            <Constraint>
                <Name NameId="ATCCM">Are timestamp cryptographic constraints
met?</Name>
                <Status>OK</Status>
                <AdditionalInfo>Validation time : 2020-01-23 13:09 for token with ID :
[T-1F9F5B92B9DF5678CD57A7B47FEEADDE365017BD67E5E7468A5ADF14F6CC901F]</AdditionalInfo>
            </Constraint>
            <Conclusion>
                <Indication>PASSED</Indication>
            </Conclusion>
            <CryptographicInfo>
                <Algorithm>http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-
MGF1</Algorithm>
                <KeyLength>2048</KeyLength>
                <Secure>true</Secure>
                <NotAfter>2022-12-31T23:00:00</NotAfter>
            </CryptographicInfo>
        </SAV>
        <CertificateChain>
            <ChainItem Id="C-
9ACF4D4E37AD647B1310D55800234BBF17FFFE01CAF7E63ED7223B770CB0D2E9">
                <Source>TRUSTED_LIST</Source>
            </ChainItem>
            <ChainItem Id="C-
5F522A439611DEC501F0468FF6A455D23DFBD08EEB33964A0C5EDE95593E1FA8">
                <Source>TIMESTAMP</Source>
            </ChainItem>
            <ChainItem Id="C-
4D24807B9CAD5110F40ED79D934346D7C9B0290431DC9B11A40BBB86FCF2AEF6">
                <Source>TIMESTAMP</Source>
            </ChainItem>
        </CertificateChain>
        <Conclusion>
            <Indication>PASSED</Indication>
        </Conclusion>
    </BasicBuildingBlocks>
    <TLAnalysis CountryCode="EU" URL="https://ec.europa.eu/tools/lotl/eu-lotl.xml" Id
="LOTL-EC2AE37FE9A43B48B1CFE2A57EBEE2BD6373EDFF36537EB1BC905747ACBF4C3B" Title="List
Of Trusted Lists EU">
        <Constraint>
            <Name NameId="QUAL_TL_FRESH">Is the trusted list fresh?</Name>
```

```
            <Status>WARNING</Status>
            <Warning NameId="QUAL_TL_FRESH_ANS">The trusted list is not considered as
fresh!</Warning>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_EXP">Is the trusted list not expired?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_VERSION">Does the trusted list have the expected
version?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_WS">Is the trusted list well signed?</Name>
            <Status>OK</Status>
        </Constraint>
        <Conclusion>
            <Indication>PASSED</Indication>
            <Warnings NameId="QUAL_TL_FRESH_ANS">The trusted list is not considered as
fresh!</Warnings>
        </Conclusion>
    </TLAnalysis>
    <TLAnalysis CountryCode="DE" URL="https://www.nrca-ds.de/st/TSL-XML.xml" Id="TL-
B66584674C05B72F6BB09FF13C632EB1BA72A0DD0DCF9A6822EF73374F6DFC48" Title="Trusted List
DE">
        <Constraint>
            <Name NameId="QUAL_TL_FRESH">Is the trusted list fresh?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_EXP">Is the trusted list not expired?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_VERSION">Does the trusted list have the expected
version?</Name>
            <Status>OK</Status>
        </Constraint>
        <Constraint>
            <Name NameId="QUAL_TL_WS">Is the trusted list well signed?</Name>
            <Status>OK</Status>
        </Constraint>
        <Conclusion>
            <Indication>PASSED</Indication>
        </Conclusion>
    </TLAnalysis>
</DetailedReport>
```

Available implementations of DSSDocument

DSS allows creation of different kinds of DSSDocument :

- `InMemoryDocument` : fully loads in memory. This type of DSSDocument can be instantiated with an array of bytes, an InputStream,...
- `FileDocument` : refers an existing File
- `DigestDocument` : only contains pre-computed digest values for a given document. That allows a user to avoid sending the full document (detached signatures).

DigestDocument

```
// Firstly, we load a basic DSSDocument (FileDocument or InMemoryDocument)
DSSDocument fileDocument = new FileDocument("src/main/resources/xml_example.xml");

// After that, we create a DigestDocument
DigestDocument digestDocument = new DigestDocument(DigestAlgorithm.SHA1, fileDocument
.getDigest(DigestAlgorithm.SHA1));
digestDocument.setName(fileDocument.getName());

// We can add an additional needed digest value(s). Eg : for a SHA-256 based signature
digestDocument.addDigest(DigestAlgorithm.SHA256, fileDocument.getDigest
(DigestAlgorithm.SHA256));
```

Management of signature tokens

The DSS framework is able to create signatures from PKCS#11, PKCS#12 and MS CAPI. Java 6 is inherently capable of communicating with these kinds of KeyStores. To be independent of the signing media, DSS framework uses an interface named SignatureTokenConnection to manage different implementations of the signing process. The base implementation is able to sign a stream of the data in one step. That means that all the data to be signed needs to be sent to the SSCD. This is the case for MS CAPI. As to the PKCS#11 and PKCS#12, which give to the developer a finer control in the signature operation, the DSS framework implements the AsyncSignatureTokenConnection abstract class that permits to execute the digest operation and signature operation in two different threads or even two different hardwares.

This design permits also other card providers/adopters to create own implementations. For example, this can be used for a direct connection to the Smartcard through Java 6 PC/SC.

PKCS#11

PKCS#11 is widely used to access smart cards and HSMs. Most commercial software uses PKCS#11 to access the signature key of the CA or to enrol user certificates. In the DSS framework, this standard is encapsulated in the class Pkcs11SignatureToken.

Pkcs11SignatureToken usage

```
try (Pkcs11SignatureToken token = new Pkcs11SignatureToken("C:\\Windows\\System32
\\beidpkcs11.dll")) {

    List<DSSPrivateKeyEntry> keys = token.getKeys();
    for (DSSPrivateKeyEntry entry : keys) {
        System.out.println(entry.getCertificate().getCertificate());
    }

    ToBeSigned toBeSigned = new ToBeSigned("Hello world".getBytes());
    SignatureValue signatureValue = token.sign(toBeSigned, DigestAlgorithm.SHA256,
keys.get(0));

    System.out.println("Signature value : " + Utils.toBase64(signatureValue.getValue(
)));
}
```

PKCS#12

This standard defines a file format commonly used to store the private key and corresponding public key certificate protecting them by password.

In order to use this format with the DSS framework you have to go through the class Pkcs12SignatureToken.

Pkcs12SignatureToken usage

```
try (Pkcs12SignatureToken token = new Pkcs12SignatureToken(
"src/main/resources/user_a_rsa.p12", new PasswordProtection("password".toCharArray())
)) {

    List<DSSPrivateKeyEntry> keys = token.getKeys();
    for (DSSPrivateKeyEntry entry : keys) {
        System.out.println(entry.getCertificate().getCertificate());
    }

    ToBeSigned toBeSigned = new ToBeSigned("Hello world".getBytes());
    SignatureValue signatureValue = token.sign(toBeSigned, DigestAlgorithm.SHA256,
keys.get(0));

    System.out.println("Signature value : " + Utils.toBase64(signatureValue.getValue(
)));
}
```

MS CAPI

If the middleware for communicating with an SSDC provides a CSP based on MS CAPI specification,

then to sign the documents you can use MSCAPISignatureToken class.

MSCAPISignatureToken usage

```
try (MSCAPISignatureToken token = new MSCAPISignatureToken()) {

    List<DSSPrivateKeyEntry> keys = token.getKeys();
    for (DSSPrivateKeyEntry entry : keys) {
        System.out.println(entry.getCertificate().getCertificate());
    }

    ToBeSigned toBeSigned = new ToBeSigned("Hello world".getBytes());
    SignatureValue signatureValue = token.sign(toBeSigned, DigestAlgorithm.SHA256,
keys.get(0));

    System.out.println("Signature value : " + Utils.toBase64(signatureValue.getValue(
)));
}
```

Other Implementations

As you can see, it is easy to add another implementation of the SignatureTokenConnection, thus enabling the framework to use other API than the provided three (PKCS#11, PKCS#12 and MS CAPI). For example, it is likely that in the future PC/SC will be the preferred way of accessing a Smartcard. Although PKCS#11 is currently the most used API, DSS framework is extensible and can use PC/SC. For our design example we propose to use PC/SC to communicate with the Smartcard.

Management of certificates sources

The validation of a certificate requires the access to some other certificates from multiple sources like trusted lists, trust store, the signature itself: certificates can be contained inside or any other source. Within the framework, an X509 certificate is modelled through the class:

- `eu.europa.esig.dss.x509.CertificateToken`

This encapsulation helps make certificate handling more suited to the needs of the validation in the context of trust. Each certificate is unambiguously identified by its issuer DN and serial number. The framework associates a unique internal identifier to each certificate but this identifier is not calculated on the data contained in the certificate and therefore varies from one application to another. However, it is independent of its source. It allows comparison of certificates issued by different sources. Certificate tokens are grouped into pools. A certificate token can be declared in several pools. The class that models a pool is called:

- `eu.europa.esig.dss.x509.CertificatePool`

This class allows keeping only one occurrence of the certificate in the given context (i.e. validation).

The CertificateSource interface provides abstraction for accessing a certificate, regardless of the

source. However, each source has its own type:

- `eu.europa.esig.dss.x509.CertificateSourceType`

This information is used, for example, to distinguish between the certificate from a trusted source and the others. A source has one and only one type, but a certificate token can be found in multiple sources. The DSS framework supplies some standard implementations, but also gives the possibility to implement owner solutions. Among the standard solutions you can find:

- `eu.europa.esig.dss.x509.CommonCertificateSource`

This is the superclass of almost of the certificate sources. It implements the common method CommonCertificateSource#get returns the list of CertificateToken(s) corresponding to the given subject distinguished name. Note that the content of the encapsulated certificates pool can be different from the content of the source. Only CertificateToken(s) present in the source are taken into account. It exposes also the method CommonCertificateSource#addCertificate which gives the possibility to add manually any X509Certificate as a part of this source and as a part of the encapsulated pool. If the certificate is already present in the pool its source type is associated to the token.

- `eu.europa.esig.dss.x509.SignatureCertificateSource`

Some certificate sources are based on data encapsulated within a signature. That means that the set of certificates is available and the software only needs to find the certificate using its subject name. This class also contains a list of methods allowing to obtain specialized list of certificates contained in the source by their original location.

- `eu.europa.esig.dss.tsl.TrustedListsCertificateSource`

Certificates coming from the list of Trusted Lists. This class gives the mechanism to define the set of trusted certificates (trust anchors). They are used in the validation process to decide if the prospective certificate chain has a trust anchor. See chapter TLValidationJob to get more information about trusted lists loading (e.g. EU Trusted List).

Management of CRL and OCSP sources

A CRL is a time-stamped list identifying revoked certificates. It is signed by a Certificate Authority (CA) and made freely available in a public repository. Each revoked certificate is identified in a CRL by its certificate serial number.

The Online Certificate Status Protocol (OCSP) is an Internet protocol used for obtaining the revocation status of an unique X.509 digital certificate.

For every certificate, the validity has to be checked via CRL or OCSP responses. The information may originate from different CRLSources or OCSPSources: For easing the usage of such sources, DSS implements a CRLSource and OCSPSource interfaces (which inherit from RevocationSource), which offer a generic, uniform way of accessing CRL and OCSP sources. Furthermore, a caching mechanism can be easily attached to those sources, optimizing the access time to revocation information by reducing network connections to online servers.

The interface CRLSource defines the method which returns CRLToken for the given certificate/issuer certificate couple:

CRLSource usage

```
CRLToken crlToken = crlSource.getRevocationToken(certificateToken,
issuerCertificateToken);
```

The interface OCSPSource defines the method which returns OCSPToken for the given certificate/issuer certificate couple:

OCSPSource usage

```
OCSPToken ocspToken = ocspSource.getRevocationToken(certificateToken,
issuerCertificateToken);
```

We use these classes during the certificate validation process through "validationContext" object (based on ValidationContext class) which is a "cache" for one validation request that contains every object retrieved so far. This object in turn instantiates a "verifier" based on CSPAndCRLCertificateVerifier class whose role is to fetch revocation data by querying an OCSP server first and then a CRL server if no OCSP response could be retrieved. In general, we can distinguish three main sources:

- Offline sources;
- Online sources;
- Sources with the cache mechanism.

Repository Revocation Source

The above-mentioned class allows caching of CRL and OCSP responses to a user-chosen source. By default DSS provides a JDBC based implementation for this class, but other implementations also can be created. The class contains a complete set of functions to save revocation data to a database, extract, update and remove it.
Furthermore, the `Repository Revocation Source` allows the implementer to define a backup revocation source, for the case if the database does not contains the certificate's revocation data yet.

List of cached Revocation sources implemented in DSS:

- `JdbcRevocationSource`
  - `JdbcCacheCRLSource`
  - `JdbcCacheOCSPSource`

The classes allow the following configuration :

JdbcCacheCRLSource usage

```
// Creates an instance of JdbcCacheCRLSource
JdbcCacheCRLSource cacheCRLSource = new JdbcCacheCRLSource();

// Set dataSource for the repository
cacheCRLSource.setDataSource(dataSource);

// Allows definition of an alternative dataLoadet to be used to access a revocation
// from online sources if a requested revocation is not present in the repository or
has been expired (see below).
cacheCRLSource.setProxySource(onlineCRLSource);

// All setters accept values in seconds
Long oneWeek = (long) (60 * 60 * 24 * 7); // seconds * minutes * hours * days

// If "nextUpdate" field is not defined for a revocation token, the value of
"defaultNextUpdateDelay"
// will be used in order to determine when a new revocation data should be requested.
// If the current time is not beyond the "thisUpdate" time + "defaultNextUpdateDelay",
// then a revocation data will be retrieved from the repository source, otherwise a
new revocation data
// will be requested from a proxiedSource.
// Default : null (a new revocation data will be requested of "nestUpdate" field is
not defined).
cacheCRLSource.setDefaultNextUpdateDelay(oneWeek);

// Defines a custom maximum possible nextUpdate delay. Allows limiting of a time
interval
// from "thisUpdate" to "nextUpdate" defined in a revocation data.
// Default : null (not specified, the "nextUpdate" value provided in a revocation is
used).
cacheCRLSource.setMaxNextUpdateDelay(oneWeek); // force refresh every week (eg : ARL)

// Defines if a revocation should be removed on its expiration.
// Default : true (removes revocation from a repository if expired).
cacheCRLSource.setRemoveExpired(true);

// Creates an SQL table
cacheCRLSource.initTable();

// Extract CRL for a certificate
RevocationToken crlRevocationToken = cacheCRLSource.getRevocationToken
(certificateToken, issuerCertificateToken);
```

And an example for JdbcCacheOCSPSource :

JdbcCacheOCSPSource usage

```
JdbcCacheOCSPSource cacheOCSPSource = new JdbcCacheOCSPSource();
cacheOCSPSource.setDataSource(dataSource);
cacheOCSPSource.setProxySource(onlineOCSPSource);
Long threeMinutes = (long) (60 * 3);
cacheOCSPSource.setDefaultNextUpdateDelay(threeMinutes); // default nextUpdateDelay
(if not defined in the revocation data)
cacheOCSPSource.initTable();
RevocationToken ocspRevocationToken = cacheOCSPSource.getRevocationToken
(certificateToken, certificateToken);
```

Be aware that you have to initialize a table before start of working with the cached revocation repository.

Other implementations of CRL and OCSP Sources

Such sources find the status of a certificate either from a list stored locally or using the information contained in the advanced signature or online way. Here is the list of sources already implemented in the DSS framework:

- CRL sources
  - `OfflineCRLSource` : This class that implements in a generic way the findCrl method that operates on the different CRLs implemented in children classes.
    - `ListCRLSource` : This source maintains a list of CRLToken.
    - `SignatureCRLSource` : The advanced signature contains a list of CRL that was needed to validate the signature. This class is a basic skeleton that is able to retrieve the needed CRL from a list. The child needs to retrieve the list of wrapped CRLs.
      - `CAdESCRLSource` : Retrieves information from a CAdES signature.
      - `PAdESCRLSource` : Retrieves information from a PAdES signature.
      - `XAdESCRLSource` : Retrieves information from a XAdES signature.
      - `ExternalResourcesCRLSource` : A class that can instantiate a list of certificate revocation lists from a directory where should be the individual lists (each individual list file must end with the extension ".crl").
  - `OnlineCRLSource` : Retrieves CRL files from online sources.
  - `JdbcCacheCrlSource` : Implementation of the `JdbcRevocationSource`. This implementation allows storage of valid CRL entries to a defined `DataSource'` and retrieve them locally.
- OCSP sources
  - `OfflineOCSPSource` : An abstract class that helps to implement OCSPSource with an already loaded list of OCSPToken. It implements in a generic way the getOCSPResponse method that operates on the different OCSP implementations in children classes.
    - `ListOCSPSource` : Implements an OCSPSource from a list of OCSPToken.
    - `SignatureOCSPSource` : The advanced signature contains a list of OCSPResp that was

needed to validate the signature. This class is a basic skeleton that is able to retrieve the needed OCSPResp from a list. The children need to retrieve the list of wrapped OCSPResp.

- `CAdESOCSPSource` : Retrieves information from a CAdES signature.
- `PAdESOCSPSource` : Retrieves information from a PAdES signature.
- `XAdESOCSPSource` : Retrieves information from a XAdES signature.
- `ExternalResourcesOCSPSource` : A class that can instantiate a list of OCSPToken from a directory where should be the individual DER Encoded X509 certificates files (each individual file must end with the extension ".der").

◦ `OnlineOCSPSource` : Retrieves OCSP responses from online source.

◦ `JdbcCacheOcspSource` : Implementation of the `JdbcRevocationSource`. This implementation allows storage of valid OCSP entries to a defined `DataSource` and retrieve them locally.

Online CRL Source

This is a representation of an Online CRL repository. This implementation will contact using HTTP protocol the CRL Responder to download the CRLs from the given URI. Note that certificate"s Authority Information Access (AIA) extension is used to find issuer's resources location like CRL file and/or Online Certificate Status Protocol (OCSP). The URIs of CRL server will be extracted from this property (OID value: 1.3.6.1.5.5.7.48.1.3).

It allows the following configuration :

OnlineCRLSource usage

```
// Instantiates a new OnlineCRLSource
OnlineCRLSource onlineCRLSource = new OnlineCRLSource();

// Allows setting an implementation of `DataLoader` interface,
// processing a querying of a remote revocation server.
// `CommonsDataLoader` instance is used by default.
onlineCRLSource.setDataLoader(new CommonsDataLoader());

// Aets a preferred protocol that will be used for obtaining a CRL.
// E.g. for a list of urls with protocols HTTP, LDAP and FTP, with a defined preferred
protocol as FTP,
// the FTP url will be called first, and in case of an unsuccessful result other url
calls will follow.
// Default : null (urls will be called in a provided order).
onlineCRLSource.setPreferredProtocol(Protocol.FTP);
```

Online OCSP Source

This is a representation of an Online OCSP repository. This implementation will contact using HTTP protocol the OCSP Responder to retrieve the OCSP response. Note that certificate's Authority Information Access (AIA) extension is used to find issuer's resources location like CRT file and/or

Online Certificate Status Protocol (OCSP). The URIs of OCSP server will be extracted from this property (OID value: 1.3.6.1.5.5.7.48.1).

It allows the following configuration :

OnlineOCSPSource usage

```
// Instantiates a new OnlineOCSPSource object
OnlineOCSPSource onlineOCSPSource = new OnlineOCSPSource();

// Allows setting an implementation of `DataLoader` interface,
// processing a querying of a remote revocation server.
// `CommonsDataLoader` instance is used by default.
onlineOCSPSource.setDataLoader(new OCSPDataLoader());

// Defines an arbitrary integer used in OCSP source querying in order to prevent a
replay attack.
// Default : null (not used by default).
onlineOCSPSource.setNonceSource(new SecureRandomNonceSource());

// Defines a DigestAlgorithm being used to generate a CertificateID in order to
complete an OCSP request.
// OCSP servers supporting multiple hash functions may produce a revocation response
// with a digest algorithm depending on the provided CertificateID's algorithm.
// Default : SHA1 (as a mandatory requirement to be implemented by OCSP servers. See
RFC 5019).
onlineOCSPSource.setCertIDDigestAlgorithm(DigestAlgorithm.SHA1);
```

CertificateVerifier configuration

The CertificateVerifier and its implementation CommonCertificateVerifier determines how DSS accesses the external resources and how it should react in some occasions. This configuration is used in both extension and validation mode.

CertificateVerifier usage

```
CertificateVerifier cv = new CommonCertificateVerifier();

// This data loader is used to collect certificates from external resources
// (AIA)
cv.setDataLoader(dataLoader);

// This certificate source is used to provide missing intermediate certificates
// (not trusted certificates)
cv.setAdjunctCertSource(adjunctCertSource);

// This certificate source is used to provide trusted certificates (the trust
// anchors where the certificate chain building should stop)
cv.setTrustedCertSource(trustedCertSource);
```

```
// The CRL Source to be used for external accesses (can be configured with a
// cache,...)
cv.setCrlSource(crlSource);

// The OCSP Source to be used for external accesses (can be configured with a
// cache,...)
cv.setOcspSource(ocspSource);

// Sets the default digest algorithm that will be used for digest calculation
// of tokens used during the validation process.
// The values will be used in validation reports.
// Default : DigestAlgorithm.SHA256
cv.setDefaultDigestAlgorithm(DigestAlgorithm.SHA512);

// Define the behavior to be followed by DSS in case of revocation checking for
// certificates issued from an unsure source (DSS v5.4+)
// Default : revocation check is disabled for unsure sources (security reasons)
cv.setCheckRevocationForUntrustedChains(false);

// DSS v5.4+ : The 3 below configurations concern the extension mode (LT/LTA
// extension)

// DSS throws an exception by default in case of missing revocation data
// Default : true
cv.setExceptionOnMissingRevocationData(true);

// DSS throws an exception if a TSU certificate chain is not covered with a
// revocation data (timestamp generation time > CRL/OCSP production time).
// Default : false
cv.setExceptionOnUncoveredPOE(true);

// DSS interrupts by default the extension process if a revoked certificate is
// present
// Default : true
cv.setExceptionOnRevokedCertificate(true);

// DSS stops the extension process if an invalid timestamp is met
// Default : true
cv.setExceptionOnInvalidTimestamp(true);

// DSS v5.5+ : throw an exception in case if there is no valid revocation data
// with thisUpdate time after the best signature time
// Example: if a signature was extended to T level then the obtained revocation
// must have thisUpdate time after production time of the signature timestamp.
// Default : false
cv.setExceptionOnNoRevocationAfterBestSignatureTime(true);

// DSS v5.4+ : defines if binary of certificates used during validation must be
included
// to produced validation reports. If false only digests will be included.
```

```
// Default : false
cv.setIncludeCertificateRevocationValues(true);

// DSS v5.4+ : defines if binary of revocation data used during validation must be
included
// to produced validation reports. If false only digests will be included.
// Default : false
cv.setIncludeCertificateRevocationValues(true);

// DSS v5.4+ : defines if binary of timestamps present into the signature must be
included
// to produced validation reports. If false only digests will be included.
// Default : false
cv.setIncludeTimestampTokenValues(true);
```

Trust Anchor(s) configuration

Trust anchors represent an important part in the signature creation / validation. That defines which are the trusted entities, which signatures can be trusted,... Do I trust certificates/signatures from another company / country / ... ?

Since the version 5.6, DSS allows to configure one or more trusted certificate source(s). These sources can be configured from a TrustStore (kind of keystore which only contains certificates), a trusted list and/or a list of trusted lists.

Multiple trusted certificate sources usage

```
CertificateVerifier cv = new CommonCertificateVerifier();
cv.setTrustedCertSources(trustStoreSource(), trustedListSource());
```

Trust store initialization

If you have a collection of certificates to trust, the easier way to provide them to DSS it to use a KeyStore / TrustStore.

Trust anchor initialization from a Trust Store

```
public CertificateSource trustStoreSource() throws IOException {
    KeyStoreCertificateSource keystore = new KeyStoreCertificateSource(new File(
"src/main/resources/keystore.p12"), "PKCS12", getPassword());

    CommonTrustedCertificateSource trustedCertificateSource = new
CommonTrustedCertificateSource();
    trustedCertificateSource.importAsTrusted(keystore);

    // Optionally, certificates can also be directly added
    trustedCertificateSource.addCertificate(DSSUtils
.loadCertificateFromBase64EncodedString(

"MIIC9TCCAd2gAwIBAgIBAjANBgkqhkiG9w0BAQUFADArMQswCQYDVQQGEwJBQTEMMAoGA1UEChMDRFNTMQ4wD
AYDVQQDEwVJQ0EgQTAeFw0xMzEyMDIxNzMzMTBaFw0xNTEyMDIxNzMzMTBaMDAxCzAJBgNVBAYTAkFBMQwwCgY
DVQQKEwNEU1MxEzARBgNVBAMTCnVzZXIgQSBSU0EwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAJUHHAphm
SDdQ1t62tppK+dLTANsE2nAj+HCpasS3ohlBsrhteRsvTAbrDyIzCmTYWu/nVI4TGvbzBESwV/QitlkoMLpYFw
32MIBf2DLmECzGJ3vm5haw6u8S9quR1h8Vu7QWd+5KMabZuR+j91RiSuoY0xS2ZQxJw1vhvW9hRYjAgMBAAGjg
aIwgZ8wCQYDVR0TBAIwADAdBgNVHQ4EFgQU9ESnTWfwg13c3LQZzqqwibY5WVYwUwYDVR0jBEwwSoAUIO1CDsB
SUcEoFZxKaWf1PAL1U+uhL6QtMCsxDDAKBgNVBAoTA0RTUzELMAkGA1UEBhMCQUExDjAMBgNVBAMTBVJDQSBBg
gEBMAsGA1UdDwQEAwIHgDARBgNVHSAECjAIMAYGBFUdIAAwDQYJKoZIhvcNAQEFBQADggEBAGnhhnoyVUhDnr/
BSbZ/uWfSuwzFPG+2V9K6WxdIaaXOORFGIdFwGlAwA/Qzpq9snfBxuTkAykxq0uEDhHTj0qXxWRjQ+Dop/Drmc
coF/zDvgGusyY1YXaABd/kc3IYt7ns7z3tpiqIz4A7a/UHplBRXfqjyaZurZuJQRaSdxh6CNhdEUiUBxkbb1Sd
MjuOgjzSDjcDjcegjvDquMKdDetvtu2Qh4ConBBo3fUImwiFRWnbudS5H2HE18ikC7gY/QIuNr7USf1PNyUgcG
2g31cMtemj7UTBHZ2V/jPf7ZXqwfnVSaYkNvM3weAI6R3PI0STjdxN6a9qjt9xld40YEdw="));

    return trustedCertificateSource;
}
```

To generate the trust store, there's an utility class CreateKeyStoreApp in the dss-cookbook module.

Trusted List Certificate Source

In several countries, a list of Trust Service Providers (TSP) is published. This list is usually published in a machine processable format (XML) and sometimes in a human-readable format (PDF). A standard (ETSI TS 119 612) exists with the specifications for the XML format.

DSS contains all needed resources to download, parse, validate and interpret the trusted list contents. Since DSS 5.6, that's possible to configure one or more independent trusted list(s) (aka not linked to a list of trusted lists) and/or one or more list of trusted lists.

If you want to collect your trusted certificates from trusted list(s), the TrustedListsCertificateSource is required. The trusted list(s) loading can require some times (connection time-out, xml parsing, xml validation,...). This process is usually executed in background. An instance of TrustedListsCertificateSource needs to be created. That will be synchronized with the TLValidationJob.

Trusted List Certificate Source

```
public CertificateSource trustedListSource() {
    return new TrustedListsCertificateSource();
}
```

TLValidationJob

The TLValidationJob allows to download, parse, validate the Trusted List(s) and Lists Of Trusted Lists (LOTL). Once the task is done, its result is stored in the TrustedListsCertificateSource. The job uses 3 different caches (download, parsing and validation) and a state-machine to be efficient.

Trusted lists are stored on the file system. That offers the possibility to run in offline mode with the stored trusted lists. Trusted Lists can be loaded from the file system and/or from Internet.

In the next sections the different configurations will be covered.

TLSource and LOTLSource

Several TLSources and several LOTLSources can be injected in a TLValidationJob. The only constraint is the uniqueness of the Trusted List URLs.

Multiple TLSources and multiple LOTLSources configuration

```
TLValidationJob validationJob = new TLValidationJob();
// Specify where is the TL/LOTL is hosted and which are the signing certificate(s) for
these TL/LOTL.
validationJob.setTrustedListSources(boliviaTLSource(), costaRicaTLSource());
validationJob.setListOfTrustedListSources(europeanLOTLSource(),
unitedStatesLOTLSource());
```

Trusted List Source (TLSource)

A TLSource allows to quickly setup a trusted list configuration. The URL and the signing certificates for this TL are mandatory. Optionally, predicates / filters can be configured to retrieve only a part of the trust service providers or trust services.

TLSource configuration

```
TLSource tlSource = new TLSource();

// Mandatory : The url where the TL needs to be downloaded
tlSource.setUrl("http://www.ssi.gouv.fr/eidas/TL-FR.xml");

// A certificate source which contains the signing certificate(s) for the
// current trusted list
tlSource.setCertificateSource(getSigningCertificatesForFrenchTL());

// Optional : predicate to filter trust services which are/were granted or
// equivalent (pre/post eIDAS).
// Input : implementation of TrustServicePredicate interface.
// Default : none (select all)
tlSource.setTrustServicePredicate(new GrantedTrustService());

// Optional : predicate to filter the trust service providers
// Input : implementation of TrustServiceProviderPredicate interface.
// Default : none (select all)
tlSource.setTrustServiceProviderPredicate(new CryptologOnlyTrustServiceProvider());

//instance of CertificateSource where all trusted certificates and their properties
(service type,...) are stored.
tlValidationJob.setTrustedListSources(tlSource);
```

List Of Trusted Lists Source (LOTLSource)

A similar configuration is possible for Lists Of Trusted Lists (LOTL). That requires an URL and the possible LOTL signers. Some other parameters are possible. By default, all listed trusted lists are loaded.

LOTLSource configuration

```
LOTLSource lotlSource = new LOTLSource();

// Mandatory : The url where the LOTL needs to be downloaded
lotlSource.setUrl("https://ec.europa.eu/tools/lotl/eu-lotl.xml");

// A certificate source which contains the signing certificate(s) for the
// current list of trusted lists
lotlSource.setCertificateSource(getSigningCertificatesForEuropeanLOTL());

// true or false for the pivot support. Default = false
// More information :
// https://ec.europa.eu/tools/lotl/pivot-lotl-explanation.html
lotlSource.setPivotSupport(true);

// Optional : the predicate which allows to find the LOTL definition in the LOTL
// Input : implementation of Predicate<OtherTSLPointerType> interface (e.g.
```

```
OtherTSLPointerPredicate)
// Default : European configuration
lotlSource.setLotlPredicate(new EULOTLOtherTSLPointer().and(new XMLOtherTSLPointer())
);

// Optional : the predicate which allows to find and/or filter the TL
// definitions in the LOTL
// Input : implementation of Predicate<OtherTSLPointerType> interface (e.g.
OtherTSLPointerPredicate)
// Default : all found trusted lists in the European LOTL
lotlSource.setTlPredicate(new EUTLOtherTSLPointer().and(new XMLOtherTSLPointer()));

// Optional : a predicate which allows to find back the signing certificates for
// the current LOTL
// Input : implementation of LOTLSigningCertificatesAnnouncementSchemeInformationURI
interface.
// Default : not defined
//
// OfficialJournalSchemeInformationURI allows to specify the Official Journal
// URL where are published the signing certificates
lotlSource.setSigningCertificatesAnnouncementPredicate(
        new OfficialJournalSchemeInformationURI("https://eur-lex.europa.eu/legal-
content/EN/TXT/?uri=uriserv:OJ.C_.2019.276.01.0001.01.ENG"));

// Optional : predicate to filter trust services which are/were granted or
// equivalent (pre/post eIDAS). This parameter is applied on the related trusted
// lists
// Input : implementation of TrustServicePredicate interface.
// Default : none (select all)
lotlSource.setTrustServicePredicate(new GrantedTrustService());

// Optional : predicate to filter the trust service providers. This parameter is
// applied on the related trusted lists
// Input : implementation of TrustServiceProviderPredicate interface.
// Default : none (select all)
lotlSource.setTrustServiceProviderPredicate(new CryptologOnlyTrustServiceProvider());

tlValidationJob.setListOfTrustedListSources(lotlSource);
```

DSSFileLoader

The FileCacheDataLoader is used to download the trusted list contents on the file-system. Two different configurations are needed. Both of them share the same folder :

- offline refresh : disabled download from Internet and unlimited cache expiration
- online refresh : enabled download from Internet and limited cache expiration

Offline and Online refresh configuration

```
public DSSFileLoader offlineLoader() {
    FileCacheDataLoader offlineFileLoader = new FileCacheDataLoader();
    offlineFileLoader.setCacheExpirationTime(Long.MAX_VALUE);
    offlineFileLoader.setDataLoader(new IgnoreDataLoader()); // do not download from
Internet
    offlineFileLoader.setFileCacheDirectory(tlCacheDirectory());
    return offlineFileLoader;
}

public DSSFileLoader onlineLoader() {
    FileCacheDataLoader offlineFileLoader = new FileCacheDataLoader();
    offlineFileLoader.setCacheExpirationTime(0);
    offlineFileLoader.setDataLoader(dataLoader()); // instance of DataLoader which can
access to Internet (proxy,...)
    offlineFileLoader.setFileCacheDirectory(tlCacheDirectory());
    return offlineFileLoader;
}
```

The SynchronizationStrategy

The SynchronizationStrategy defines which are the trusted lists or list of trusted lists to be synchronized. By default, DSS synchronizes all of them. DSS don't reject any expired / invalid /... trusted lists. The content is trusted and a warning is added in a signature / certificate validation.

The strategy is configurable via the interface SynchronizationStrategy :

Example of a custom SynchronizationStrategy

```
public SynchronizationStrategy allValidTrustedListsStrategy() {

    return new SynchronizationStrategy() {

        @Override
        public boolean canBeSynchronized(TLInfo trustedList) {
            return trustedList.getValidationCacheInfo().isValid();
        }

        @Override
        public boolean canBeSynchronized(LOTLInfo listOfTrustedList) {
            return listOfTrustedList.getValidationCacheInfo().isValid();
        }

    };

}
```

DSS provides two implementations : ExpirationAndSignatureCheckStrategy and AcceptAllStrategy

(default).

The CacheCleaner

The CacheCleaner specifies how DSS clear the cache in case of expired URL,... 2 options are available : memory and file-system.

CacheCleaner Configuration

```
public CacheCleaner cacheCleaner() {
    CacheCleaner cacheCleaner = new CacheCleaner();

    cacheCleaner.setCleanMemory(true); // free the space in memory

    cacheCleaner.setCleanFileSystem(true); // remove the stored file(s) on the file-
system

    // if the file-system cleaner is enabled, inject the configured loader from the
    // online or offline refresh data loader.
    cacheCleaner.setDSSFileLoader(offlineLoader());

    return cacheCleaner;
}
```

Alerting from TL Loading

Since the version 5.6 of DSS, that's possible to launch alerts in some situations (eg : invalid TL signature, LOTL location change,...). Alert works with two concepts : detection and alert handler. After the download/parsing/validation and before the synchronization, the results are tested to detect events and launch alert(s).

Examples of Alerting

```
TLValidationJob job = new TLValidationJob();
// ...

// Add a log message in case of invalid signatures
TLAlert tlBrokenSignatureAlert = new TLAlert(new TLSignatureErrorDetection(), new
LogTLSignatureErrorAlertHandler());

// Send an email in case of new Official Journal detected
AlertHandler<LOTLInfo> mailSender = new AlertHandler<LOTLInfo>() {

    @Override
    public void alert(LOTLInfo currentInfo) {
        String newOJUrl = currentInfo.getParsingCacheInfo()
.getSigningCertificateAnnouncementUrl();
        // code to send an email
        SampleUtils.sendEmail(newOJUrl);
```

```
    }

};

// The europeanLOTLSource is configured with an
// OfficialJournalSchemeInformationURI
LOTLAlert officialJournalDesynchronizationAlert = new LOTLAlert(new
OJUrlChangeDetection(europeanLOTLSource()), mailSender);

// Update a database in case of LOTL location change
AlertHandler<LOTLInfo> databaseUpgrader = new AlertHandler<LOTLInfo>() {

    @Override
    public void alert(LOTLInfo currentInfo) {
        String newLOTLUrl = null;

        String currentLOTLUrl = currentInfo.getUrl();
        List<PivotInfo> pivots = currentInfo.getPivotInfos();
        for (PivotInfo pivot : pivots) {
            if (!Utils.areStringsEqual(currentLOTLUrl, pivot.getLOTLLocation())) {
                newLOTLUrl = pivot.getLOTLLocation();
                break;
            }
        }

        // code to update a database
        SampleUtils.updateDatabase(newLOTLUrl);
    }

};
LOTLAlert lotlLocationChangeAlert = new LOTLAlert(new LOTLLocationChangeDetection
(europeanLOTLSource()), databaseUpgrader);

// add all alerts on the job
job.setAlerts(Arrays.asList(tlBrokenSignatureAlert,
officialJournalDesynchronizationAlert, lotlLocationChangeAlert));
```

Executor Service

An Executor Service parameter allows you to customize a way of the program execution on your Java machine, by configuring a number of possible threads to be running, await time and so on.

Executor Service

```
// Allows configuration of the execution process
// Default : Executors.newCachedThreadPool() is used
tlValidationJob.setExecutorService(Executors.newSingleThreadExecutor());
```

Complete configuration for the European LOTL

Below, you can find a complete configuration for the European List Of Trusted Lists. The URLs need to be externalized.

European LOTL Configuration

```
// Should be externalized
private static final String LOTL_URL = "https://ec.europa.eu/tools/lotl/eu-lotl.xml";
private static final String OJ_URL = "https://eur-lex.europa.eu/legal-
content/EN/TXT/?uri=uriserv:OJ.C_.2019.276.01.0001.01.ENG";

public TLValidationJob job() {
    TLValidationJob job = new TLValidationJob();
    job.setOfflineDataLoader(offlineLoader());
    job.setOnlineDataLoader(onlineLoader());
    job.setTrustedListCertificateSource(trustedCertificateSource());
    job.setSynchronizationStrategy(new AcceptAllStrategy());
    job.setCacheCleaner(cacheCleaner());

    LOTLSource europeanLOTL = europeanLOTL();
    job.setListOfTrustedListSources(europeanLOTL);

    job.setAlerts(Arrays.asList(tlSigningAlert(), tlExpirationDetection(), ojUrlAlert
(europeanLOTL),
            lotlLocationAlert(europeanLOTL)));

    return job;
}

public TrustedListsCertificateSource trustedCertificateSource() {
    return new TrustedListsCertificateSource();
}

public LOTLSource europeanLOTL() {
    LOTLSource lotlSource = new LOTLSource();
    lotlSource.setUrl(LOTL_URL);
    lotlSource.setCertificateSource(officialJournalContentKeyStore());
    lotlSource.setSigningCertificatesAnnouncementPredicate(new
OfficialJournalSchemeInformationURI(OJ_URL));
    lotlSource.setPivotSupport(true);
    return lotlSource;
}

public CertificateSource officialJournalContentKeyStore() {
    try {
        return new KeyStoreCertificateSource(new File("
src/main/resources/keystore.p12"), "PKCS12", "dss-password");
    } catch (IOException e) {
        throw new DSSException("Unable to load the keystore", e);
    }
```

```
}

public DSSFileLoader offlineLoader() {
    FileCacheDataLoader offlineFileLoader = new FileCacheDataLoader();
    offlineFileLoader.setCacheExpirationTime(Long.MAX_VALUE);
    offlineFileLoader.setDataLoader(new IgnoreDataLoader());
    offlineFileLoader.setFileCacheDirectory(tlCacheDirectory());
    return offlineFileLoader;
}

public DSSFileLoader onlineLoader() {
    FileCacheDataLoader offlineFileLoader = new FileCacheDataLoader();
    offlineFileLoader.setCacheExpirationTime(0);
    offlineFileLoader.setDataLoader(dataLoader());
    offlineFileLoader.setFileCacheDirectory(tlCacheDirectory());
    return offlineFileLoader;
}

public File tlCacheDirectory() {
    File rootFolder = new File(System.getProperty("java.io.tmpdir"));
    File tslCache = new File(rootFolder, "dss-tsl-loader");
    if (tslCache.mkdirs()) {
        LOG.info("TL Cache folder : {}", tslCache.getAbsolutePath());
    }
    return tslCache;
}

public CommonsDataLoader dataLoader() {
    return new CommonsDataLoader();
}

public CacheCleaner cacheCleaner() {
    CacheCleaner cacheCleaner = new CacheCleaner();
    cacheCleaner.setCleanMemory(true);
    cacheCleaner.setCleanFileSystem(true);
    cacheCleaner.setDSSFileLoader(offlineLoader());
    return cacheCleaner;
}

// Optionally : alerting.
// Recommended detections : OJUrlChangeDetection + LOTLLocationChangeDetection

public TLAlert tlSigningAlert() {
    TLSignatureErrorDetection signingDetection = new TLSignatureErrorDetection();
    LogTLSignatureErrorAlertHandler handler = new LogTLSignatureErrorAlertHandler();
    return new TLAlert(signingDetection, handler);
}

public TLAlert tlExpirationDetection() {
    TLExpirationDetection expirationDetection = new TLExpirationDetection();
    LogTLExpirationAlertHandler handler = new LogTLExpirationAlertHandler();
```

```
    return new TLAlert(expirationDetection, handler);
}

public LOTLAlert ojUrlAlert(LOTLSource source) {
    OJUrlChangeDetection ojUrlDetection = new OJUrlChangeDetection(source);
    LogOJUrlChangeAlertHandler handler = new LogOJUrlChangeAlertHandler();
    return new LOTLAlert(ojUrlDetection, handler);
}

public LOTLAlert lotlLocationAlert(LOTLSource source) {
    LOTLLocationChangeDetection lotlLocationDetection = new
LOTLLocationChangeDetection(source);
    LogLOTLLocationChangeAlertHandler handler = new LogLOTLLocationChangeAlertHandler
();
    return new LOTLAlert(lotlLocationDetection, handler);
}
```

The TL / LOTL refresh

The TL / LOTL loading in DSS works as below :

- Download / parse / validate all LOTLSources from the configuration with/without pivot support (multi-threaded)
- Analyze introduced changes and expire cache entries (new TL URLs, new signing certificates for a TL,...)
- Create TLSources from the retrieved LOTLs
- Combine these TLSources with independent TLSources (from the configuration)
- Download / parse / validate all TLs (multi-threaded)
- If alerts are configured, test if an alert needs to be launched
- If the debug is enabled, print in the log the cache status
- Synchronize the TrustedListCertificateSource
- If the cache cleaner is configured, execute it
- If the debug is enabled, print in the log the cache status

The refresh can be called with the offline or the online loader and run exactly the same code

How to refresh the Trusted List(s) and Lists of Trusted Lists

```
TLValidationJob validationJob = new TLValidationJob();

// call with the Offline Loader (application initialization)
validationJob.offlineRefresh();

// call with the Online Loader (callable every day/hour in a cron)
validationJob.onlineRefresh();
```

Java Keystore Management

Generally (like in case of European LOTL) DSS downloads Trusted Lists by using the SSL protocol (for resources using HTTPS extension), that requires to have a certificate of a remote source in the Java trust store. The certificates have their own validity period and can expire. If a certificated is expired, it will be replaced on a server by a new one in order to support a secure SSL connection. The easiest way to know if your Java trust store is outdated and new certificates need to be added is to check your logs during a TLValidationJob update :

```
ERROR 14052 --- [pool-2-thread-30] e.e.e.dss.tsl.runnable.AbstractAnalysis  : Unable
to process GET call for url [https://sr.riik.ee/tsl/estonian-tsl.xml]. Reason : [PKIX
path building failed: sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target]
```

The `SunCertPathBuilderException` means that the certificate established the secure connection is not trusted by your Java Virtual Machine. In order to add the certificate to the trust store, you need to do the following steps (the example is based on Windows OS and Google Chrome browser):

1. Open the failed URL in your browser. In our case it will be 'https://sr.riik.ee/tsl/estonian-tsl.xml' obtained from the logs.
2. Click on a lock icon next to the URL in the tab you just opened. It will open a window about the current connection status.
3. Click on 'Certificate' button to open the Certificate window.
4. Go to 'Details' tab and choose 'Copy to File...'.
5. Process the 'Certificate Export Wizard', by saving the certificate in one of '.CER' formats. Store the file in your file system. For us it will create a file 'ee.cer'.
6. Run 'Command Prompt' with administrator permissions (right click → 'Run As Administrator').
7. Execute the following line (ensure that 'keytool' is installed) :

Certificate import

```
keytool -import -alias newCert -file pathToCert\ee.cer -keystore pathToJavaDirectory
\lib\security\cacerts -storepass changeit
```

The default password for a Java keystore is "changeit". Ensure that you have a default configuration, or use another password you have configured.

In order to apply changes, the application using Java must be rebooted.

After these steps the `TLValidationJob` will successfully download the target Trusted List (i.e. Estonian in our example).

This described algorithm is not only one available solution, if you have difficulties with this, you can search in the Internet for another working for you solution.

TLValidationJobSummary

The class TLValidationJobSummary contains all processed data about the download (time, error,...), the parsing (extracted information, parsing error,...) and the signature validation (signing certificate, signing time,...).

How to retrieve the information about the TLValidationJob process

```
TrustedListsCertificateSource trustedListCertificateSource = new
TrustedListsCertificateSource();

TLValidationJob job = new TLValidationJob();
job.setTrustedListCertificateSource(trustedListCertificateSource);

// ... config & refresh ...

// A cache content summary can be computed on request
TLValidationJobSummary summary = job.getSummary();

// All information about processed LOTLSources
List<LOTLInfo> lotlInfos = summary.getLOTLInfos();
LOTLInfo lotlInfo = lotlInfos.get(0);
// All data about the download (last occurrence, cache status, error,...)
DownloadInfoRecord downloadCacheInfo = lotlInfo.getDownloadCacheInfo();

// All data about the parsing (date, extracted data, cache status,...)
ParsingInfoRecord parsingCacheInfo = lotlInfo.getParsingCacheInfo();

// All data about the signature validation (signing certificate, validation
// result, cache status,...)
ValidationInfoRecord validationCacheInfo = lotlInfo.getValidationCacheInfo();

// All information about processed TLSources (which are not linked to a
// LOTLSource)
List<TLInfo> otherTLInfos = summary.getOtherTLInfos();

// or the last update can be collected from the TrustedListsCertificateSource
TLValidationJobSummary lastSynchronizedSummary = trustedListCertificateSource
.getSummary();
```

TSP Sources

The Time Stamp Authority by creating time-stamp tokens provides independent and irrefutable proof of time for business transactions, e-documents and digital signatures. The TSA must comply with the IETF RFC 3161 specifications (cf. [R07]). A time-stamp is obtained by sending the digest value of the given data and digest algorithm to the Time Stamp Authority. The returned time-stamp is a signed data that contains the digest value, the identity of the TSA, and the time of stamping. This proves that the given data existed before the time of stamping. The DSS framework proposes

TSPSource interface to implement the communication with TSA. The class OnlineTSPSource is the default implementation of TSP using HTTP(S) communication layer. The following bit of Java code illustrates how you might use this class:

OnlineTSPSource usage

```
final String tspServer = "http://dss.nowina.lu/pki-factory/tsa/good-tsa";
OnlineTSPSource tspSource = new OnlineTSPSource(tspServer);
tspSource.setDataLoader(new TimestampDataLoader()); // uses the specific content-type

final DigestAlgorithm digestAlgorithm = DigestAlgorithm.SHA256;
final byte[] toDigest = "Hello world".getBytes("UTF-8");
final byte[] digestValue = DSSUtils.digest(digestAlgorithm, toDigest);
final TimestampBinary tsBinary = tspSource.getTimeStampResponse(digestAlgorithm,
digestValue);

LOG.info(DSSUtils.toHex(tsBinary.getBytes()));
```

Time-stamp policy

A time-stamp policy is a "named set of rules that indicates the applicability of a time-stamp token to a particular community and/or class of application with common security requirements". A TSA may define its own policy which enhances the policy defined in RFC 3628. Such a policy shall incorporate or further constrain the requirements identified in RFC 3628. A time-stamp policy may be defined by the user of times-stamp services.

Composite TSP Source

Sometimes, timestamping servers may encounter interruptions (restart,...). To avoid failing signature extension, DSS allows a user to configure several TSP Sources. DSS will try source by source until getting an usable timestamp token.

Configuration of a CompositeTSPSource

```
// Create a map with several TSPSources
TimestampDataLoader timestampDataLoader = new TimestampDataLoader();// uses the
specific content-type

OnlineTSPSource tsa1 = new OnlineTSPSource("http://dss.nowina.lu/pki-factory/tsa/ee-
good-tsa");
tsa1.setDataLoader(timestampDataLoader);
OnlineTSPSource tsa2 = new OnlineTSPSource("http://dss.nowina.lu/pki-factory/tsa/good-
tsa");
tsa2.setDataLoader(timestampDataLoader);

Map<String, TSPSource> tspSources = new HashMap<>();
tspSources.put("TSA1", tsa1);
tspSources.put("TSA2", tsa2);

// Instantiate a new CompositeTSPSource and set the different sources
CompositeTSPSource tspSource = new CompositeTSPSource();
tspSource.setTspSources(tspSources);

final DigestAlgorithm digestAlgorithm = DigestAlgorithm.SHA256;
final byte[] toDigest = "Hello world".getBytes("UTF-8");
final byte[] digestValue = DSSUtils.digest(digestAlgorithm, toDigest);

// DSS will request the tsp sources (one by one) until getting a valid token.
// If none of them succeed, a DSSException is thrown.
final TimestampBinary tsBinary = tspSource.getTimeStampResponse(digestAlgorithm,
digestValue);

LOG.info(DSSUtils.toHex(tsBinary.getBytes()));
```

Supported algorithms

DSS supports several signature algorithms (combination of an encryption algorithm and a digest algorithm). Below, you can find the supported combinations. The support of the algorithms depends on the registered OID (ASN1) or URI (XML).

In the next table, XAdES also applies to ASiC with embedded XAdES signatures and CAdES also concerns PAdES and ASiC with embedded CAdES signatures.

SmartCards/HSMs don't allow signing with all digest algorithms. Please refer to your SmartCard/HSM provider.

Table 7. Supported algorithms

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512	SHA3-224	SHA3-256	SHA3-384	SHA3-512	MD2	MD5	RIPEMD160
RSA												
XAdES	✓	✓	✓	✓	✓						✓	✓
CAdES	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RSA-PSS												
XAdES	✓	✓	✓	✓	✓	✓	✓	✓	✓			
CAdES	✓	✓	✓	✓	✓	✓	✓	✓	✓			
ECDSA												
XAdES	✓	✓	✓	✓	✓							✓
CAdES	✓	✓	✓	✓	✓	✓	✓	✓	✓			
DSA												
XAdES	✓		✓									
CAdES	✓	✓	✓	✓	✓	✓	✓	✓	✓			
HMAC												
XAdES	✓	✓	✓	✓	✓							✓
CAdES	✓	✓	✓	✓	✓	✓	✓	✓	✓			

Multi-threading

DSS can be used in multi-threaded environments but some points need to be considered like resources sharing and caching. All operations are stateless and this fact requires to be maintained. Some resources can be shared, others are proper to an operation.

For each provided operation, DSS requires a CertificateVerifier object. This object is responsible to provide certificates and accesses to external resources (AIA, CRL, OCSP,...). At the beginning of all operation, a new internal CertificatePool is created and all available certificates are copied. Throughout the signature/validation process, the CertificatePool content evolves. Certificates are added/updated from the signature, timestamp(s), revocation data,... Revocation data / issuer certificates are collected and added to the certificate. Certificate status are updated to give as much as possible information. For these reasons, integrators need to be careful about the CertificateVerifier configuration.

Resource sharing

The trusted certificates can be shared between multiple threads because these certificates are static. This means they don't require more analysis. Their status won't evolve. For these certificates, DSS doesn't need to collect issuer certificate and/or their revocation data.

In opposition, the adjunct certificates cannot be shared. These certificates concern a specific

signature/validation operation. This parameter is used to provide missing certificate(s). When DSS is unable to build the complete certificate path with the provided certificates (as signature parameters or embedded within a signature), it is possible to inject not present certificates. These certificates are not necessarily trusted and may require future "modifications" like revocation data collection,...

Caching

In case of multi-threading usage, we strongly recommend caching of external resources. All external resources can be cached (AIA, CRL, OCSP) to improve performances and to avoid requesting too much time the same resources. FileCacheDataLoader and JdbcCacheCRLSource can help you in this way.

JAXB modules

Since the version 5.5, DSS provides the following JAXB modules with a harmonized structure :

- `dss-policy-jaxb` - defines validation policy JAXB model
- `dss-diagnostic-jaxb` - defines Diagnostic Data JAXB model
- `dss-detailed-report-jaxb` - defines Detailed Report JAXB model
- `dss-simple-report-jaxb` - defines Simple Report JAXB model
- `dss-simple-certificate-report-jaxb` - defines Certificate Simple Report JAXB model

All modules share the same logic and have the following structure (where *** is a model name):

dss-*-jaxb**

src/main/java

eu.europa.esig.dss.*

- ***.java** - wrapper(s) which eases the JAXB manipulation
- **...**
- ***Facade.java** - class which allows marshalling/unmarshalling of jaxb objects, generation of HTML/PDF content, etc.
- ***XmlDefiner.java** - class which contains the model definition (XSD, XSLT references, ObjectFactory)
- **jaxb** - generated on compile time
  - **Xml***.java** - JAXB model
  - **...**

src/main/resources

xsd

- ***.xsd** - XML Schema (XSD) for the Detailed Report model
- **binding.xml** - XJC instructions to generate the JAXB model from the XSD

- **xslt**
  - **html**
    - ***.xslt** - XML Stylesheet for the HTML generation
  - **pdf**
    - ***.xslt** - XML Stylesheet for the PDF generation

In the main classes, a `Facade` is present to quickly operate with the JAXB objects (eg: marshall, unmarshall, generate the HTML/PDF, validate the XML structure,...).

DetailedReportFacade usage

```
Reports completeReports = documentValidator.validateDocument();

DetailedReportFacade detailedReportFacade = DetailedReportFacade.newFacade();

// Transforms the JAXB object to String (xml content)
String marshalledDetailedReport = detailedReportFacade.marshall(completeReports
.getDetailedReportJaxb());

// Transforms the String (xml content) to a JAXB Object
XmlDetailedReport xmlDetailedReport = detailedReportFacade.unmarshall
(marshalledDetailedReport);

// Generates the HTML content for the given Detailed Report (compatible with
// BootStrap)
// Similar method is available for PDF generation (requires Apache FOP)
String htmlDetailedReport = detailedReportFacade.generateHtmlReport(completeReports
.getDetailedReportJaxb());
```

A `XmlDefiner` is also available with the access to the embedded XML Schemas (XSD), the XML Stylesheets (XSLT) to be able to generate the HTML or the PDF content (for DSS specific JAXB) and the JAXB Object Factory.

DetailedReportXmlDefiner usage

```
// The JAXB Object Factory
ObjectFactory objectFactory = DetailedReportXmlDefiner.OBJECT_FACTORY;

// The JAXBContext (cached)
JAXBContext jaxbContext = DetailedReportXmlDefiner.getJAXBContext();

// The XML Schema to validate a XML content (cached)
Schema schema = DetailedReportXmlDefiner.getSchema();

// The Templates object with the loaded XML Stylesheet to generate the HTML
// content from the JAXB Object (cached)
Templates bootstrap3Templates = DetailedReportXmlDefiner.getHtmlBootstrap3Templates();

// The Templates object with the loaded XML Stylesheet to generate the PDF
// content from the JAXB Object (cached)
Templates pdfTemplates = DetailedReportXmlDefiner.getPdfTemplates();
```

Report stylesheets

The report modules (namely: `dss-simple-report-jaxb`, `dss-simple-certificate-report-jaxb` and `dss-detailed-report-jaxb`) contain two XSLT style sheets for final reports generation:

- Bootstrap 3 XSLT;
- Bootstrap 4 XSLT.

By default, in DSS since version 5.6 the style sheet for Bootstrap 4 is used. In order to generate a report with a selected style sheet you need to call a relevant method in a Facade class (see classes definition above):

Bootstrap 3 generation

```
String bootstrap3Report = SimpleReportFacade.newFacade().generateHtmlBootstrap3Report
(xmlSimpleReport);
```

Otherwise, in case if you need to customize the transformer, you can create a report by using an XmlDefiner:

Bootstrap 3 custom generation

```
try (Writer writer = new StringWriter()) {
    Transformer transformer = SimpleCertificateReportXmlDefiner
.getHtmlBootstrap3Templates().newTransformer();
    // specify custom parameters if needed
    transformer.transform(new StreamSource(new StringReader(simpleReport)), new
StreamResult(writer));
    String bootstrap3Report = writer.toString();
}
```

I18N (Internationalization)

Since DSS 5.6 a new module has been introduced allowing changing of a language for reports generated by DSS. The current version of the framework allows customization of text values only for a `DetailedReport`.

A target language of the report can be set with the following code:

Language customization

```
SignedDocumentValidator validator = SignedDocumentValidator.fromDocument
(signedDocument);
// A target Locale must be defined for the validator
validator.setLocale(Locale.FRENCH); // for French language
```

In case if no language is specified, the framework will use a default Locale obtained from OS on a running machine. If a requested language is not found, a default translation will be used.

As a default configuration DSS provides English translation.

In order to provide a custom translation, a new file must be created inside `src\main\resources` directory of your project with a name followed by one of the patterns:

`dss-messages_XX.properties` or `dss-messages_XX_YY.properties`, where:

- XX - an abbreviation of a target language;
- YY - a country code.

For example, for a French language a file with a name `dss-messages_fr.properties` need to be created, or `dss-messages_fr_FR.properties` to use it only in France local.

Additional features

Certificate validation

DSS offers the possibility to validate a certificate. For a given certificate, the framework builds a

certificate path until a known trust anchor (trusted list, keystore,...), validates each found certificate (OCSP / CRL) and determines its European "qualification".

To determine the certificate qualification, DSS follows the draft standard ETSI TS 119 172-4 ([R09]). It analyses the certificate properties (QCStatements, Certificate Policies,...) and applies possible overrules from the related trusted list ("catched" qualifiers from a trust service). More information about qualifiers can be found in the standard ETSI TS 119 612 ([R10]).

DSS always computes the status at 2 different times : certificate issuance and signing/validation time. The certificate qualification can evolve in the time, its status is not immutable (eg: a trust service provider lost its granted status). The eIDAS regulation ([R11]) clearly defines these different times in the Article 32 and related Annex I.

Validate a certificate and retrieve its qualification level

```
// Firstly, we load the certificate to be validated
CertificateToken token = DSSUtils.loadCertificate(new File(
"src/main/resources/keystore/ec.europa.eu.1.cer"));

// We need a certificate verifier and configure it  (see specific chapter about the
CertificateVerifier configuration)
CertificateVerifier cv = new CommonCertificateVerifier();

// We create an instance of the CertificateValidator with the certificate
CertificateValidator validator = CertificateValidator.fromCertificate(token);
validator.setCertificateVerifier(cv);

// We execute the validation
CertificateReports certificateReports = validator.validate();

// We have 3 reports
// The diagnostic data which contains all used and static data
DiagnosticData diagnosticData = certificateReports.getDiagnosticData();

// The detailed report which is the result of the process of the diagnostic data and
the validation policy
DetailedReport detailedReport = certificateReports.getDetailedReport();

// The simple report is a summary of the detailed report or diagnostic data (more
user-friendly)
SimpleCertificateReport simpleReport = certificateReports.getSimpleReport();
```

Extract the signed data from a signature

DSS is able to retrieve the original data from a valid signature.

Retrieve original data from a signed document

```
// We have our signed document, we want to retrieve the original/signed data
DSSDocument signedDocument = new FileDocument("src/test/resources/signedXmlXadesB.xml
");

// We create an instance of DocumentValidator. DSS automatically selects the validator
depending of the
// signature file
SignedDocumentValidator documentValidator = SignedDocumentValidator.fromDocument
(signedDocument);

// We set a certificate verifier. It handles the certificate pool, allows to check the
certificate status,...
documentValidator.setCertificateVerifier(new CommonCertificateVerifier());

// We retrieve the found signatures
List<AdvancedSignature> signatures = documentValidator.getSignatures();

// We select the wanted signature (the first one in our current case)
AdvancedSignature advancedSignature = signatures.get(0);

// We call get original document with the related signature id (DSS unique ID)
List<DSSDocument> originalDocuments = documentValidator.getOriginalDocuments
(advancedSignature.getId());

// We can have one or more original documents depending of the signature (ASiC,
PDF,...)
DSSDocument original = originalDocuments.get(0);

original.save("target/original.xml");
```

REST and SOAP Services

DSS offers REST and SOAP web services. Additionally, we also provide a SOAP-UI project and Postman samples in the `dss-cookbook` module.

The different webservices are :

- Signature webservices (`dss-signature-soap` / `dss-signature-rest`) and their clients : they expose methods to allow signing or extending a signature from a client.
- Server-signing webservice (`dss-server-signing-soap` / `dss-server-signing-rest`) and their clients : they expose method to retrieve keys from a server (PKCS#11, PKCS#12, HSM,...) and to sign the digest on the server side.
- Signature validation webservices (`dss-validation-soap` / `dss-validation-rest`) and their clients : they expose methods to allow signature validation, with an optional detached file and an optional validation policy.
- Certificate validation webservices (`dss-certificate-validation-soap` / `dss-certificate-`

validation-rest) and their clients : they expose methods to allow certificate validation, with an optional provided certificate chain and custom validation time.

- Timestamp webservices (dss-timestamp-remote-soap / dss-timestamp-remote-rest) and their clients : they expose methos to allow remote timestamp creation, by providing digest value to be timestamped and a digest algorithm, used for the digets calculation.

The data structure in webservices is similar in both REST and SOAP modules.

The documentation will covers the REST calls. All the REST services present in DSS are compliant with OpenAPI Specification.

REST signature service

This service exposes 4 methods for one or more document(s) :

Rest signature service

```
// Initializes the rest client
RestDocumentSignatureService restClient = new RestDocumentSignatureServiceImpl();

// Defines RemoteSignatureParameters
RemoteSignatureParameters parameters = new RemoteSignatureParameters();
parameters.setSignatureLevel(SignatureLevel.PAdES_BASELINE_B);
parameters.setSigningCertificate(new RemoteCertificate(privateKey.getCertificate()
.getEncoded()));
parameters.setSignaturePackaging(SignaturePackaging.ENVELOPING);
parameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// Initialize a RemoteDocument object to be signed
FileDocument fileToSign = new FileDocument(new File("src/test/resources/sample.xml"));
RemoteDocument toSignDocument = new RemoteDocument(Utils.toByteArray(fileToSign
.openStream()), fileToSign.getName());

// computes the digest to be signed
ToBeSignedDTO dataToSign = restClient.getDataToSign(new DataToSignOneDocumentDTO
(toSignDocument, parameters));

// Creates a SignOneDocumentDTO
SignatureValue signatureValue = signingToken.sign(DTOConverter.toToBeSigned(
dataToSign), DigestAlgorithm.SHA256, privateKey);
SignOneDocumentDTO signDocument = new SignOneDocumentDTO(toSignDocument, parameters,
        new SignatureValueDTO(signatureValue.getAlgorithm(), signatureValue.getValue(
)));

// Adds the signature value to the document
RemoteDocument signedDocument = restClient.signDocument(signDocument);

// Define the extention parameters
RemoteSignatureParameters extendParameters = new RemoteSignatureParameters();
extendParameters.setSignatureLevel(SignatureLevel.PAdES_BASELINE_T);
```

```
// Extends the existing signature
RemoteDocument extendedDocument = restClient.extendDocument(new ExtendDocumentDTO
(signedDocument, extendParameters));

// Defines timestamp parameters
RemoteTimestampParameters remoteTimestampParameters = new RemoteTimestampParameters();
remoteTimestampParameters.setDigestAlgorithm(DigestAlgorithm.SHA256);

// Defines a Timestamp document DTO
TimestampOneDocumentDTO timestampOneDocumentDTO = new TimestampOneDocumentDTO
(extendedDocument, remoteTimestampParameters);

// Timestamps a provided document (available for PDF, ASiC-E and ASiC-S container
formats)
RemoteDocument timestampedDocument = restClient.timestampDocument
(timestampOneDocumentDTO);
```

Get data to sign

The method allows retrieving the data to be signed. The user sends the document to be signed, the parameters (signature level,...) and the certificate chain.

The parameters in getDataToSign and signDocument MUST be the same (especially the signing date).

Request

```
POST /services/rest/signature/one-document/getDataToSign HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 2753

{
  "parameters" : {
    "signingCertificate" : {
      "encodedCertificate" :
"MIIC6jCCAdKgAwIBAgIGLtYU17tXMA0GCSqGSIb3DQEBCwUAMDAxGzAZBgNVBAMMElJvb3RTZWxmU2lnbmVkR
mFrZTERMA8GA1UECgwIRFNTLXRlc3QwHhcNMTcwNjA4MTEyNjAxWhcNNDcwNzA0MDc1NzI0WjAoMRMwEQYDVQQ
DDApTaWduZXJGYWtlMREwDwYDVQQKDAhEU1MtdGVzdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
MI3kZhtnipn+iiZHZ9ax8FlfE5Ow/cFwBTfAEb3R1ZQUp6/BQnBt7Oo0JWBtc9qkv7JUDdcBJXPV5QWS5AyMPH
pqQ75Hitjsq/Fzu8eHtkKpFizcxGa9BZdkQjh4rSrtO1Kjs0Rd5DQtWSgkeVCCN09kN0ZsZ0ENY+Ip8QxSmyzt
sStkYXdULqpwz4JEXW9vz64eTbde4vQJ6pjHGarJf1gQNEc2XzhmI/prXLysWNqC7lZg7PUZUTrdegABTUzYCR
J1kWBRPm4qo0LN405c94QQd45a5kTgowHzEgLnAQI28x0M3A59TKC+ieNc6VF1PsTLpUw7PNI2VstX5jAuasCA
wEAAaMSMBAwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBCwUAA4IBAQCK6LGA01TR+rmU8p6yhAi4OkDN2b1
dbIL8l8iCMYopLCxx8xqq3ubZCOxqh1X2j6pgWzarb0b/MUix00IoUvNbFOxAW7PBZIKDLnm6LsckRxs1U32sC
9d1LOHe3WKBNB6GZALT1ewjh7hSbWjftlmcovq+6eVGA5cvf2u/2+TkKkyHV/NR394nXrdsdpvygwypEtXjetz
D7UT93Nuw3xcV8VIftIvHf9LjU7h+UjGmKXG9c15eYr3SzUmv6kyOI0Bvw14PWtsWGl0QdOSRvIBBrP4adCnGT
gjgjk9LTcO8B8FKrr+8lHGuc0bp4lIUToiUkGILXsiEeEg9WAqm+XqO"
```

```
    },
    "certificateChain" : [ ],
    "detachedContents" : null,
    "asicContainerType" : null,
    "signatureLevel" : "CAdES_BASELINE_B",
    "signaturePackaging" : "ENVELOPING",
    "signatureAlgorithm" : "RSA_SHA256",
    "digestAlgorithm" : "SHA256",
    "encryptionAlgorithm" : "RSA",
    "referenceDigestAlgorithm" : null,
    "contentTimestampParameters" : {
      "digestAlgorithm" : "SHA256",
      "canonicalizationMethod" : "http://www.w3.org/2001/10/xml-exc-c14n#"
    },
    "signatureTimestampParameters" : {
      "digestAlgorithm" : "SHA256",
      "canonicalizationMethod" : "http://www.w3.org/2001/10/xml-exc-c14n#"
    },
    "archiveTimestampParameters" : {
      "digestAlgorithm" : "SHA256",
      "canonicalizationMethod" : "http://www.w3.org/2001/10/xml-exc-c14n#"
    },
    "signWithExpiredCertificate" : false,
    "generateTBSWithoutCertificate" : false,
    "blevelParams" : {
      "trustAnchorBPPolicy" : true,
      "signingDate" : 1566540526950,
      "claimedSignerRoles" : null,
      "policyId" : null,
      "policyQualifier" : null,
      "policyDescription" : null,
      "policyDigestAlgorithm" : null,
      "policyDigestValue" : null,
      "policySpuri" : null,
      "commitmentTypeIndications" : null,
      "signerLocationPostalAddress" : [ ],
      "signerLocationPostalCode" : null,
      "signerLocationLocality" : null,
      "signerLocationStateOrProvince" : null,
      "signerLocationCountry" : null,
      "signerLocationStreet" : null
    }
  },
  "toSignDocument" : {
    "bytes" : "SGVsbG8=",
    "digestAlgorithm" : null,
    "name" : "RemoteDocument"
  }
}
```

Response

```
HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 06:08:46 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
Transfer-Encoding: chunked
Content-Length: 448

{
  "bytes" :
"MYIBOzARBgsqhkiG9w0BCRACDzECBQAwFQYLKoZIhvcNAQkQAhExBjAEogIwADAYBgkqhkiG9w0BCQMxCwYJK
oZIhvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0xOTA4MjMwNjA4NDZaMC0GCSqGSIb3DQEJNDEgMB4wDQYJYIZIAWU
DBAIBBQChDQYJKoZIhvcNAQELBQAwLwYJKoZIhvcNAQkEMSIEIBhfjbMicf4l9WGm/JOLLiZDBuwwTtpRgAfRd
kgmOBlpMHcGCyqGSIb3DQEJEAIvMWgwZjBkMGIEIALz68oBYydCU7yAnSdJjdQbsDFtfmsGaWARXeFVWJ2cMD4
wNKQyMDAxGzAZBgNVBAMMElJvb3RTZWxmU2lnbmVkRmFrZTERMA8GA1UECgwIRFNTLXRlc3QCBi7WFNe7Vw=="
}
```

Sign document

The method allows generation of the signed document with the received signature value.

The parameters in getDataToSign and signDocument MUST be the same (especially the signing date).

Request

```
POST /services/rest/signature/one-document/signDocument HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 2842

{
  "parameters" : {
    "signingCertificate" : {
      "encodedCertificate" :
"MIIC6jCCAdKgAwIBAgIGLtYU17tXMA0GCSqGSIb3DQEBCwUAMDAxGzAZBgNVBAMMElJvb3RTZWxmU2lnbmVkR
mFrZTERMA8GA1UECgwIRFNTLXRlc3QwHhcNMTcwNjA4MTEyNjAxWhcNNDcwNzA0MDc1NzI0WjAoMRMwEQYDVQQ
DDApTaWduZXJGYWtlMREwDwYDVQQKDAhEU1MtdGVzdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
MI3kZhtnipn+iiZHZ9ax8FlfE5Ow/cFwBTfAEb3R1ZQUp6/BQnBt7Oo0JWBtc9qkv7JUDdcBJXPV5QWS5AyMPH
pqQ75Hitjsq/Fzu8eHtkKpFizcxGa9BZdkQjh4rSrtO1Kjs0Rd5DQtWSgkeVCCN09kN0ZsZ0ENY+Ip8QxSmyzt
sStkYXdULqpwz4JEXW9vz64eTbde4vQJ6pjHGarJf1gQNEc2XzhmI/prXLysWNqC7lZg7PUZUTrdegABTUzYCR
J1kWBRPm4qo0LN405c94QQd45a5kTgowHzEgLnAQI28x0M3A59TKC+ieNc6VF1PsTLpUw7PNI2VstX5jAuasCA
```

```
wEAAaMSMBAwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBCwUAA4IBAQCK6LGA01TR+rmU8p6yhAi4OkDN2b1
dbIL8l8iCMYopLCxx8xqq3ubZCOxqh1X2j6pgWzarb0b/MUix00IoUvNbFOxAW7PBZIKDLnm6LsckRxs1U32sC
9d1LOHe3WKBNB6GZALT1ewjh7hSbWjftlmcovq+6eVGA5cvf2u/2+TkKkyHV/NR394nXrdsdpvygwypEtXjetz
D7UT93Nuw3xcV8VIftIvHf9LjU7h+UjGmKXG9c15eYr3SzUmv6kyOI0Bvw14PWtsWGl0QdOSRvIBBrP4adCnGT
gjgjk9LTcO8B8FKrr+8lHGuc0bp4lIUToiUkGILXsiEeEg9WAqm+XqO"
    },
    "certificateChain" : [ ],
    "detachedContents" : null,
    "asicContainerType" : null,
    "signatureLevel" : "CAdES_BASELINE_B",
    "signaturePackaging" : "ENVELOPING",
    "signatureAlgorithm" : "RSA_SHA256",
    "digestAlgorithm" : "SHA256",
    "encryptionAlgorithm" : "RSA",
    "referenceDigestAlgorithm" : null,
    "contentTimestampParameters" : {
      "digestAlgorithm" : "SHA256",
      "canonicalizationMethod" : "http://www.w3.org/2001/10/xml-exc-c14n#"
    },
    "signatureTimestampParameters" : {
      "digestAlgorithm" : "SHA256",
      "canonicalizationMethod" : "http://www.w3.org/2001/10/xml-exc-c14n#"
    },
    "archiveTimestampParameters" : {
      "digestAlgorithm" : "SHA256",
      "canonicalizationMethod" : "http://www.w3.org/2001/10/xml-exc-c14n#"
    },
    "signWithExpiredCertificate" : false,
    "generateTBSWithoutCertificate" : false,
    "blevelParams" : {
      "trustAnchorBPPolicy" : true,
      "signingDate" : 1566540526716,
      "claimedSignerRoles" : null,
      "policyId" : null,
      "policyQualifier" : null,
      "policyDescription" : null,
      "policyDigestAlgorithm" : null,
      "policyDigestValue" : null,
      "policySpuri" : null,
      "commitmentTypeIndications" : null,
      "signerLocationPostalAddress" : [ ],
      "signerLocationPostalCode" : null,
      "signerLocationLocality" : null,
      "signerLocationStateOrProvince" : null,
      "signerLocationCountry" : null,
      "signerLocationStreet" : null
    }
  },
  "signatureValue" : {
    "algorithm" : "RSA_SHA256",
    "value" : "AQIDBA=="
```

```
  },
  "toSignDocument" : {
    "bytes" : "SGVsbG8=",
    "digestAlgorithm" : null,
    "name" : "RemoteDocument"
  }
}
```

Response

```
HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 06:08:46 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
Transfer-Encoding: chunked
Content-Length: 1769

{
  "bytes" :
"MIIE2QYJKoZIhvcNAQcCoIIEyjCCBMYCAQExDzANBglghkgBZQMEAgEFADAUBgkqhkiG9w0BBwGgBwQFSGVsb
G+gggLuMIIC6jCCAdKgAwIBAgIGLtYU17tXMA0GCSqGSIb3DQEBCwUAMDAxGzAZBgNVBAMMElJvb3RTZWxmU2l
nbmVkRmFrZTERMA8GA1UECgwIRFNTLXRlc3QwHhcNMTcwNjA4MTEyNjAxWhcNNDcwNzA0MDc1NzI0WjAoMRMwE
QYDVQQDDApTaWduZXJGYWtlMREwDwYDVQQKDAhEU1MtdGVzdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQo
CggEBAMI3kZhtnipn+iiZHZ9ax8FlfE5Ow/cFwBTfAEb3R1ZQUp6/BQnBt7Oo0JWBtc9qkv7JUDdcBJXPV5QWS
5AyMPHpqQ75Hitjsq/Fzu8eHtkKpFizcxGa9BZdkQjh4rSrtO1Kjs0Rd5DQtWSgkeVCCN09kN0ZsZ0ENY+Ip8Q
xSmyztsStkYXdULqpwz4JEXW9vz64eTbde4vQJ6pjHGarJf1gQNEc2XzhmI/prXLysWNqC7lZg7PUZUTrdegAB
TUzYCRJ1kWBRPm4qo0LN405c94QQd45a5kTgowHzEgLnAQI28x0M3A59TKC+ieNc6VF1PsTLpUw7PNI2VstX5j
AuasCAwEAAaMSMBAwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBCwUAA4IBAQCK6LGA01TR+rmU8p6yhAi4O
kDN2b1dbIL8l8iCMYopLCxx8xqq3ubZCOxqh1X2j6pgWzarb0b/MUix00IoUvNbFOxAW7PBZIKDLnm6LsckRxs
1U32sC9d1LOHe3WKBNB6GZALT1ewjh7hSbWjftlmcovq+6eVGA5cvf2u/2+TkKkyHV/NR394nXrdsdpvygwypE
tXjetzD7UT93Nuw3xcV8VIftIvHf9LjU7h+UjGmKXG9c15eYr3SzUmv6kyOI0Bvw14PWtsWGl0QdOSRvIBBrP4
adCnGTgjgjk9LTcO8B8FKrr+8lHGuc0bp4lIUToiUkGILXsiEeEg9WAqm+XqOMYIBpjCCAaICAQEwOjAwMRswG
QYDVQQDDBJSb290U2VsZlNpZ25lZEZha2UxETAPBgNVBAoMCERTUy10ZXN0AgYu1hTXu1cwDQYJYIZIAWUDBAI
BBQCgggE7MBEGCyqGSIb3DQEJEAIPMQIFADAVBgsqhkiG9w0BCRACETEGMASiAjAAMBgGCSqGSIb3DQEJAzELB
gkqhkiG9w0BBwEwHAYJKoZIhvcNAQkFMQ8XDTE5MDgyMzA2MDg0NlowLQYJKoZIhvcNAQk0MSAwHjANBglghkg
BZQMEAgEFAKENBgkqhkiG9w0BAQsFADAvBgkqhkiG9w0BCQQxIgQgGF+NsyJx/iX1Yab8k4suJkMG7DBO2lGAB
9F2SCY4GWkwdwYLKoZIhvcNAQkQAi8xaDBmMGQwYgQgAvPrygFjJ0JTvICdJ0mN1BuwMW1+awZpYBFd4VVYnZw
wPjA0pDIwMDEbMBkGA1UEAwwSUm9vdFNlbGZTaWduZWRGYWtlMREwDwYDVQQKDAhEU1MtdGVzdAIGLtYU17tXM
A0GCSqGSIb3DQEBCwUABAQBAgME",
  "digestAlgorithm" : null,
  "name" : "RemoteDocument-signed-cades-baseline-b.pkcs7"
}
```

Extend document

The method allows extension of an existing signature to a stronger level.

Request

```
POST /services/rest/signature/one-document/extendDocument HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 8885

{
  "toExtendDocument" : {
    "bytes" :
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48ZHM6U2lnbmF0dXJlIHhtbG5zOmRzPSJod
HRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjIiBJZD0iaWQtYWZkZTc4MjQzNjQ2OGRkNzRlZWIxODF
mN2NlMTEwZTEiPjxkczpTaWduZWRJbmZvPjxkczpDYW5vbmljYWxpemF0aW9uTWV0aG9kIEFsZ29yaXRobT0ia
HR0cDovL3d3dy53My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8+PGRzOlNpZ25hdHVyZU1ldGhvZCBBbGd
vcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZHNpZy1tb3JlI3JzYS1zaGEyNTYiLz48ZHM6U
mVmZXJlbmNlIElkPSJyLWlkLTEiIFR5cGU9IiIgVVJJPSJzYW1wbGUueG1sIj48ZHM6RGlnZXN0TWV0aG9kIEF
sZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhMjU2Ii8+PGRzOkRpZ2VzdFZhb
HVlPmtjREhPWmp3WmhWZnVEaHVoQ2VDRVJSbVlwVEg0Smo0Um1mVlZpMzFROWc9PC9kczpEaWdlc3RWYWx1ZT4
8L2RzOlJlZmVyZW5jZT48ZHM6UmVmZXJlbmNlIFR5cGU9Imh0dHA6Ly91cmkuZXRzaS5vcmcvMDE5MDMjU2lnb
mVkUHJvcGVydGllcyIgVVJJPSIjeGFkZXMtaWQtYWZkZTc4MjQzNjQ2OGRkNzRlZWIxODFmN2NlMTEwZTEiPjx
kczpUcmFuc2Zvcm1zPjxkczpUcmFuc2Zvcm0gQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzEwL
3htbC1leGMtYzE0biMiLz48L2RzOlRyYW5zZm9ybXM+PGRzOkRpZ2VzdE1ldGhvZCBBbGdvcml0aG09Imh0dHA
6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZW5jI3NoYTI1NiIvPjxkczpEaWdlc3RWYWx1ZT5EenR3TlRtUm9PQ
W02L2xNSThSeW01eFpQekl2TFlEem4vZWJZWWtQc3I0PTwvZHM6RGlnZXN0VmFsdWU+PC9kczpSZWZlcmVuY2U
+PC9kczpTaWduZWRJbmZvPjxkczpTaWduYXR1cmVWYWx1ZSBJZD0idmFsdWUtaWQtYWZkZTc4MjQzNjQ2OGRkN
zRlZWIxODFmN2NlMTEwZTEiPllBN3NFTnQzTjh1ZkxGTW5LcjM2cjBQcXpNaVkzUTBzKytJR1RFVUMwc3BheFV
2MGRIWk0wZC95bjNrcExKTG9Va0k0TTNmbGo1V0duODNrZjA1QnFNMWtoc1g2MUdKemFGVFBHcG03YWtSUUtod
m9IMjV5eXFUWVhFU2xCY20wNGl6aUtoTE16WmpVZng0L0IxWkl5c3Y1cElCZ0oycjJvaTZqTG9wOXd3M2dlNGM
0WUpvYUsrU1hrNmh5VE5PY044UGpHZTYzV1lPVE5WUFFGdmphOEJud2crYTBiQnV3RCs4TjZmd2lnQ2RXNWEvN
ERKVWUvSjhNYjcwWkk4UG9Pem5HRGZpK1RQYmlJZVZtQ2JsNW1Vb1VnMlEveFlsdUpmTGgzdUdRQVhLQnZGNDV
vRElIUlZlZm5OL0QvV3l0QUNsVVZEb1FTeXdlbW5qUHBxRjhlZz09PC9kczpTaWduYXR1cmVWYWx1ZT48ZHM6S
2V5SW5mbz48ZHM6WDUwOURhdGE+PGRzOlg1MDlDZXJ0aWZpY2F0ZT5NSUlEMURDQ0FyeWdBd0lCQWdJQkNqQU5
CZ2txaGtpRzl3MEJBUXNGQURCTk1SQXdEZ1lEVlFRRERBZG5iMjlrTFdOaE1Sa3dGd1lEVlFRS0RCQk9iM2RwY
m1FZ1UyOXNkWFJwYjI1ek1SRXdEd1lEVlFRTERBaFFTMGt0VkVWVFZERUxNQWtHQTFVRUJoTUNURlV3SGhjTk1
UWXhNREkyTURjMU5ETXhXaGNOTVRnd09ESTJNRGMxTkRNeFdqQlBNUkl3RUFZRFZRUUREQWxuYjI5a0xYVnpaW
El4R1RBWEJnTlZCQW9NRUU1dmQybHVZU0JUYjJ4MWRHbHZibk14RVRBUEJnTlZCQXNNQ0ZCTFNTMVVSVk5VTVF
zd0NRWURWUVFHRXdKTVZUQ0NBU0l3RFFZSktvWklodmNOQVFFQkJRQURnZ0VQQURDQ0FRb0NnZ0VCQUxSQ1VJU
VpidzNuU2RMcCtCOWN6RUNncFpra1E1eFY0ZzlNLzd3bGc5N29DQ2Y3VUVoOUJBMWQrellqc3p2K0JKMWJKWlB
nYW4yMTQ0QXZnc29HSmZiNlVJeVZXNGdrbFVnSWwxYXJVdm9uK1drS25zZUZ1UU9mSnlqU0ZVREl3bnV2cDBoe
mNKWEhYUm1MZG15aCtuKzZOTUgwb201dFZvU2ZRcnRCVmlDTGVTTVZ6dUQ1RVBqMG1JUmN4OTFwTDM4ZTNGTlR
XN05hR1pMZWV6dUZ1Ui9xN3o5M2xMa3ZaNFZBTU5HR0x2SVhPWWVSQlpNeVBocEJaNEwzQThJM0VFbEtXSC8xT
HdpaVhUVFNHMXNNNld2TVRWYmYydmJkNDduWlJRQTJtU3BOR2pRb3VPQUVyZmVWVlVxeklDZ2hRQ0hSR09OdVN
MRy9IZnFGSGI0aldnMENBd0VBQWFPQnZEQ0J1VEFPQmdOVkhROEJBZjhFQkFNQ0JrQXdnWWNHQ0NzR0FRVUZCd
0VCQkhzd2VUQTVCZ2dyQmdFRkJRY3dBWVl0YUhSMGNEb3ZMMlJ6Y3k1dWIzZHBibUV1YkhVdmNHdHBMV1poWTN
SdmNua3ZiMk56Y0M5bmIyOWtMV05oTUR3R0NDc0dBUVVGQnpBQ2hqQm9kSFJ3T2k4dlpITnpMbTV2ZDJsdVlTN
XNkUzl3YTJrdFptRmpkRzl5ZVM5amNuUXZaMjl2WkMxallTNWpjblF3SFFZRFZSME9CQllFRk4ycEhELzdQZWZ
```

```
tQlQ4b1gyOVpoV3kvT0hKMU1BMEdDU3FHU0liM0RRRUJDd1VBQTRJQkFRQkszVk9MaERJVldLb0ZycmhoV3phZ
GR0azZYUXRjd1JvTlBWU3NpL2dPcnpzZE03MEEzMXhJVHc3WWZMaHBvVkExeG83b3ZIbGRwTGxocXk5bzV3aDI
4MnlDcHFCVUF0Z3JTa0RHb2crSzdDTDZnVXBybFlpWnVHWnJ0ZzJYM2ZIUzJVc3g0WkozdElqNndWZWNERVVxS
VNGZkZUMkVzbTBRWFVuZ0lLRk1sOTVYZ210dzJ3eFhiT3pVZURkNERJUHJ2K21XNXBvQVdyNkl0c1YrSDJWUSt
aTC9rQm53V0hqU1RPYUdGaXNxWFkvYUgvMVB0QlhBKzE1K1lJV2VtSkJTdjNrRGFGek9YQUV0UjlaSThsWU9KY
XJuWTdBeS9hTjZiOXVHZmZyYm8vaFZBY0w0V0RkaGtiQk4zbTh3K2c3NkxvQVhOZUVldTA0QS8weExaelVCPC9
kczpYNTA5Q2VydGlmaWNhdGU+PGRzOlg1MDlDZXJ0aWZpY2F0ZT5NSUlENmpDQ0F0S2dBd0lCQWdJQkJEQU5CZ
2txaGtpRzl3MEJBUXNGQURCTk1SQXdEZ1lEVlFRRERBZHliMjkwTFdOaE1Sa3dGd1lEVlFRS0RCQk9iM2RwYm1
FZ1UyOXNkWFJwYjI1ek1SRXdEd1lEVlFRTERBaFFTMGt0VkVWVFZERUxNQWtHQTFVRUJoTUNURlV3SGhjTk1UW
XhNREkyTURjMU5ETXdXaGNOTVRnd09ESTJNRGMxTkRNd1dqQk5NUkF3RGdZRFZRUUREQWRuYjI5a0xXTmhNUmt
3RndZRFZRUUtEQkJPYjNkcGJtRWdVMjlzZFhScGIyNXpNUkV3RHdZRFZRUUxEQWhRUzBrdFZFVlRWREVMTUFrR
0ExVUVCaE1DVEZVd2dnRWlNQTBHQ1NxR1NJYjNEUUVCQVFVQUE0SUJEd0F3Z2dFS0FvSUJBUUNiYmw1c0tCQ2p
TQjhUTWRhY3lteC9XZk9qTVcxZ2lJalZKUlkyOEpiTldrQ1ZtdHpnbWl0Z2hmclBRUGx1ZXUwRFRhbGJEa3JTU
3loQ3Z6elBTR1B3Q0ZPYWhGL243aFFhMUYzVWFIU3hUS3JGQzVuT3dkTHp4S1JPM1dqVnRJR1JTWDJrdjFGZlV
wUXk1RXl5K3JzZlN6SjU5ZFU1WlpkV3BkYUR1RHhWVnlEZXIzRUl5Q2JHNy81SDlNRDRZdXpOcGVURldtTTZjV
VNUMDc5NlhEbGJFeFNUVEdRWEZKQTIrQ0NzeTlEWG5KYThuejBGRThmbWN2UUhlVTZrOVFicHpHak1kM0RXbEU
2bm83VWRDWUQxSDA0K3VzQnA1aGhDckFCNjcwTmRvVHJOVG1HTkFGdDRKVDB2aXRqS0hxOUtFSWQ2TGhkY20yV
Gc5M2REY1dGdEFnTUJBQUdqZ2RRd2dkRXdEZ1lEVlIwUEFRSC9CQVFEQWdlQU1FRUdBMVVkSHdRNk1EZ3dOcUE
wb0RLR01HaDBkSEE2THk5a2MzTXVibTkzYVc1aExteDFMM0JyYVMxbVlXTjBiM0o1TDJOeWJDOXliMjkwTFdOa
ExtTnliREJNQmdnckJnRUZCUWNCQVFSQU1ENHdQQVlJS3dZQkJRVUhNQUtHTUdoMGRIQTZMeTlrYzNNdWJtOTN
hVzVoTG14MUwzQnJhUzFtWVdOMGIzSjVMMk55ZEM5eWIyOTBMV05oTG1OeWREQWRCZ05WSFE0RUZnUVUrMnRGc
XBOZTNHMjNZUjh5cUJaSWlWV1MzdlV3RHdZRFZSMFRBUUgvQkFVd0F3RUIvekFOQmdrcWhraUc5dzBCQVFzRkF
BT0NBUUVBRStOdWQwNVhHT002RkVaSFdUYzgrYm16LzZCMFhRWE41NjRLV0JCaGNoOWk1R2FkanFwU3NldmtuK
3RlTHE1bTZDTG8zZTRsWDJkSjdoc1BBdnlhTHFPSXB6ZzQ5VEdkaWIxbk9CMk83NCt5QWhUOHY5Rlp0SDFFQ0h
YeFlzdXlTR0lLdmQrTDVJakpUaXMzbGw0dlU4Rkh6eVJsTTlJUW53WlI1MDZqRmNKZUdsT2d5WmgrVUxXblJOR
UV3cU44RFRGMkQwWG9nWUJzckN4Q0JqMFBwYUpGcnV2RVFxcFV1dVlnMTRSMURKRmFoTHdxVllTT0Q1Z1BobUE
wSFI0ejNHRjNqSFN6MGk5alhTVE9zVWNka3ZVSnkwdElPbnVqc1VFa2czSDZXZzNsejhUdzNJYzdWMU5IYitNQ
zVLNFp2WCs1U1l5dTArZXI3YkZzY0lyWVp3PT08L2RzOlg1MDlDZXJ0aWZpY2F0ZT48L2RzOlg1MDlEYXRhPjw
vZHM6S2V5SW5mbz48ZHM6T2JqZWN0Pjx4YWRlczpRdWFsaWZ5aW5nUHJvcGVydGllcyB4bWxuczp4YWRlcz0ia
HR0cDovL3VyaS5ldHNpLm9yZy8wMTkwMy92MS4zLjIjIiBUYXJnZXQ9IiNpZC1hZmRlNzgyNDM2NDY4ZGQ3NGV
lYjE4MWY3Y2UxMTBlMSI+PHhhZGVzOlNpZ25lZFByb3BlcnRpZXMgSWQ9InhhZGVzLWlkLWFmZGU3ODI0MzY0N
jhkZDc0ZWViMTgxZjdjZTExMGUxIj48eGFkZXM6U2lnbmVkU2lnbmF0dXJlUHJvcGVydGllcz48eGFkZXM6U2l
nbmluZ1RpbWU+MjAxNy0wOS0yOFQxMTowOTowNFo8L3hhZGVzOlNpZ25pbmdUaW1lPjx4YWRlczpTaWduaW5nQ
2VydGlmaWNhdGVWMj48eGFkZXM6Q2VydD48eGFkZXM6Q2VydERpZ2VzdD48ZHM6RGlnZXN0TWV0aG9kIEFsZ29
yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI3NoYTEiLz48ZHM6RGlnZXN0VmFsdWU+Y
ytWb2hnMGpJY1o0VVFTV2VnbENnMG9HTldzPTwvZHM6RGlnZXN0VmFsdWU+PC94YWRlczpDZXJ0RGlnZXN0Pjx
4YWRlczpJc3N1ZXJTZXJpYWxWMj5NRll3VWFSUE1FMHhFREFPQmdOVkJBTU1CMmR2YjJRdFkyRXhHVEFYQmdOV
kJBb01FRTV2ZDJsdVlTQlRiMngxZEdsdmJuTXhFVEFQQmdOVkJBc01DRkJMU1MxVVJWTlVNUXN3Q1FZRFZRUUd
Fd0pNVlFJQkNnPT08L3hhZGVzOklzc3VlclNlcmlhbFYyPjwveGFkZXM6Q2VydD48L3hhZGVzOlNpZ25pbmdDZ
XJ0aWZpY2F0ZVYyPjwveGFkZXM6U2lnbmVkU2lnbmF0dXJlUHJvcGVydGllcz48eGFkZXM6U2lnbmVkRGF0YU9
iamVjdFByb3BlcnRpZXM+PHhhZGVzOkRhdGFPYmplY3RGb3JtYXQgT2JqZWN0UmVmZXJlbmNlPSIjci1pZC0xI
j48eGFkZXM6TWltZVR5cGU+dGV4dC94bWw8L3hhZGVzOk1pbWVUeXBlPjwveGFkZXM6RGF0YU9iamVjdEZvcm1
hdD48L3hhZGVzOlNpZ25lZERhdGFPYmplY3RQcm9wZXJ0aWVzPjwveGFkZXM6U2lnbmVkUHJvcGVydGllcz48L
3hhZGVzOlF1YWxpZnlpbmdQcm9wZXJ0aWVzPjwvZHM6T2JqZWN0PjwvZHM6U2lnbmF0dXJlPg==",
    "digestAlgorithm" : null,
    "name" : "xades-detached.xml"
  },
  "parameters" : {
    "signingCertificate" : null,
    "certificateChain" : [ ],
    "detachedContents" : [ {
```

```
      "bytes" :
"77u/PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4NCjxoOnRhYmxlIHhtbG5zOmg9Imh0d
HA6Ly93d3cudzMub3JnL1RSL2h0bWw0LyI+DQoJPGg6dHI+DQoJCTxoOnRkPkhlbGxvPC9oOnRkPg0KCQk8aDp
0ZD5Xb3JsZDwvaDp0ZD4NCgk8L2g6dHI+DQo8L2g6dGFibGU+",
      "digestAlgorithm" : null,
      "name" : "sample.xml"
    } ],
    "asicContainerType" : null,
    "signatureLevel" : "XAdES_BASELINE_T",
    "signaturePackaging" : null,
    "signatureAlgorithm" : "RSA_SHA256",
    "digestAlgorithm" : "SHA256",
    "encryptionAlgorithm" : "RSA",
    "referenceDigestAlgorithm" : null,
    "contentTimestampParameters" : {
      "digestAlgorithm" : "SHA256",
      "canonicalizationMethod" : "http://www.w3.org/2001/10/xml-exc-c14n#"
    },
    "signatureTimestampParameters" : {
      "digestAlgorithm" : "SHA256",
      "canonicalizationMethod" : "http://www.w3.org/2001/10/xml-exc-c14n#"
    },
    "archiveTimestampParameters" : {
      "digestAlgorithm" : "SHA256",
      "canonicalizationMethod" : "http://www.w3.org/2001/10/xml-exc-c14n#"
    },
    "signWithExpiredCertificate" : false,
    "generateTBSWithoutCertificate" : false,
    "blevelParams" : {
      "trustAnchorBPPolicy" : true,
      "signingDate" : 1566540523973,
      "claimedSignerRoles" : null,
      "policyId" : null,
      "policyQualifier" : null,
      "policyDescription" : null,
      "policyDigestAlgorithm" : null,
      "policyDigestValue" : null,
      "policySpuri" : null,
      "commitmentTypeIndications" : null,
      "signerLocationPostalAddress" : [ ],
      "signerLocationPostalCode" : null,
      "signerLocationLocality" : null,
      "signerLocationStateOrProvince" : null,
      "signerLocationCountry" : null,
      "signerLocationStreet" : null
    }
  }
}
```

Response

```
HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 06:08:44 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
Transfer-Encoding: chunked
Content-Length: 10477

{
  "bytes" :
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiIHN0YW5kYWxvbmU9Im5vIj8+PGRzOlNpZ25hd
HVyZSB4bWxuczpkcz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnIyIgSWQ9ImlkLWFmZGU3ODI
0MzY0NjhkZDc0ZWViMTgxZjdjZTExMGUxIj48ZHM6U2lnbmVkSW5mbz48ZHM6Q2Fub25pY2FsaXphdGlvbk1ld
GhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMTAveG1sLWV4Yy1jMTRuIyIvPjxkczpTaWd
uYXR1cmVNZXRob2QgQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGRzaWctbW9yZSNyc
2Etc2hhMjU2Ii8+PGRzOlJlZmVyZW5jZSBJZD0ici1pZC0xIiBUeXBlPSIiIFVSST0ic2FtcGxlLnhtbCI+PGR
zOkRpZ2VzdE1ldGhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZW5jI3NoYTI1N
iIvPjxkczpEaWdlc3RWYWx1ZT5rY0RIT1pqd1poVmZ1RGh1aENlQ0VSUm1ZcFRINEpqNFJtZlZWaTMxUTlnPTw
vZHM6RGlnZXN0VmFsdWU+PC9kczpSZWZlcmVuY2U+PGRzOlJlZmVyZW5jZSBUeXBlPSJodHRwOi8vdXJpLmV0c
2kub3JnLzAxOTAzI1NpZ25lZFByb3BlcnRpZXMiIFVSST0iI3hhZGVzLWlkLWFmZGU3ODI0MzY0NjhkZDc0ZWV
iMTgxZjdjZTExMGUxIj48ZHM6VHJhbnNmb3Jtcz48ZHM6VHJhbnNmb3JtIEFsZ29yaXRob0iaHR0cDovL3d3d
y53My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8+PC9kczpUcmFuc2Zvcm1zPjxkczpEaWdlc3RNZXRob2Q
gQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVuYyNzaGEyNTYiLz48ZHM6RGlnZXN0V
mFsdWU+RHp0d05UbVJvT0FtNi9sTUk4UnltNXhaUHpJdkxZRHpuL2ViWVlrUHNyND08L2RzOkRpZ2VzdFZhbHV
lPjwvZHM6UmVmZXJlbmNlPjwvZHM6U2lnbmVkSW5mbz48ZHM6U2lnbmF0dXJlVmFsdWUgSWQ9InZhbHVlLWlkL
WFmZGU3ODI0MzY0NjhkZDc0ZWViMTgxZjdjZTExMGUxIj5ZQTdzRU50M044dWZMRk1uS3IzNnIwUHF6TWlZM1E
wcysrSUdURVVDMHNwYXhVdjBkSFpNMGQveW4za3BMSkxvVWtJNE0zZmxqNVdHbjgza2YwNUJxTTFraHNYNjFHS
nphRlRQR3BtN2FrUlFLaHZvSDI1eXlxVFlYRVNsQmNtMDRpemlLaExNelpqVWZ4NC9CMVpJeXN2NXBJQmdKMnI
yb2k2akxvcDl3dzNnZTRjNFlKb2FLK1NYazZoeVROT2NOOFBqR2U2M1dZT1ROVlBRRnZqYThCbndnK2EwYkJ1d
0QrOE42ZndpZ0NkVzVhLzRESlVlL0o4TWI3MFpJOFBvT3puR0RmaStUUGJpSWVWbUNibDVtVW9VZzJRL3hZbHV
KZkxoM3VHUUFYS0J2RjQ1b0RJSFJWZWZuTi9EL1d5dEFDbFVWRG9RU3l3ZW1ualBwcUY4ZWc9PTwvZHM6U2lnb
mF0dXJlVmFsdWU+PGRzOktleUluZm8+PGRzOlg1MDlEYXRhPjxkczpYNTA5Q2VydGlmaWNhdGU+TUlJRDFEQ0N
BcnlnQXdJQkFnSUJDakFOQmdrcWhraUc5dzBCQVFzRkFEQk5NUkF3RGdZRFZRUUREQWRuYjI5a0xXTmhNUmt3R
ndZRFZRUUtEQkJPYjNkcGJtRWdVMjlzZFhScGIyNXpNUkV3RHdZRFZRUUxEQWhRUzBrdFZFVlRWREVMTUFrR0E
xVUVCaE1DVEZVd0hoY05NVFl4TURJMk1EYzFORE14V2hjTk1UZ3dPREkyTURjMU5ETXhXakJQTVJJd0VBWURWU
VFEREFsbmIyOWtMWFZ6WlhJeEdUQVhCZ05WQkFvTUVFNXZkMmx1WVNCVGIyeDFkR2x2Ym5NeEVUQVBCZ05WQkF
zTUNGQkxTUzFVUlZOVU1Rc3dDUVlEVlFRR0V3Sk1WVENDQVNJd0RRWUpLb1pJaHZjTkFRRUJCUUFEZ2dFUEFEQ
0NBUW9DZ2dFQkFMUkNVSVFaYnczblNkTHArQjljekVDZ3Baa2tRNXhWNGc5TS83d2xnOTdvQ0NmN1VFaDlCQTF
kK3pZanN6ditCSjFiSlpQZ2FuMjE0NEF2Z3NvR0pmYjZVSXlWVzRna2xVZ0lsMWFyVXZvbitXa0tuc2VGdVFPZ
kp5alNGVURJd251dnAwaHpjSlhIWFJtTGRteWgrbis2Tk1IMG9tNXRWb1NmUXJ0QlZpQ0xlU01WenVENUVQajB
tSVJjeDkxcEwzOGUzRk5UVzdOYUdaTGVlenVGdVIvcTd6OTNsTGt2WjRWQU1OR0dMdklYT1llUkJaTXlQaHBCW
jRMM0E4STNFRWxLV0gvMUx3aWlYVFRTRzFzTTZXdk1UVmJmMnZiZDQ3blpSUUEybVNwTkdqUW91T0FFcmZlVlZ
VcXpJQ2doUUNIUkdPTnVTTEcvSGZxRkhiNGpXZzBDQXdFQUFhT0J2RENCdVRBT0JnTlZIUThCQWY4RUJBTUNCa
0F3Z1ljR0NDc0dBUVVGQndFQkJIc3dlVEE1QmdnckJnRUZCUWN3QVlZdGFIUjBjRG92TDJSemN5NXViM2RwYm1
FdWJIVXZjR3RwTFdaaFkzUnZjbmt2YjJOemNDOW5iMjlrTFdOaE1Ed0dDQ3NHQVFVRkJ6QUNoakJvZEhSd09pO
HZaSE56TG01dmQybHVZUzVzZFM5d2Eya3RabUZqZEc5eWVTOWpjblF2WjI5dlpDMWpZUzVqY25Rd0hRWURWUjB
```

```
PQkJZRUZOMnBIRC83UGVmbUJUOG9YMjlaaFd5L09ISjFNQTBHQ1NxR1NJYjNEUUVCQ3dVQUE0SUJBUUJLM1ZPT
GhESVZXS29GcnJoaFd6YWRkdGs2WFF0Y3dSb05QVlNzaS9nT3J6c2RNNzBBMzF4SVR3N1lmTGhwb1ZBMXhvN29
2SGxkcExsaHF5OW81d2gyODJ5Q3BxQlVBdGdyU2tER29nK0s3Q0w2Z1VwcmxZaVp1R1pydGcyWDNmSFMyVXN4N
FpKM3RJajZ3VmVjREVVcUlTRmZGVDJFc20wUVhVbmdJS0ZNbDk1WGdtdHcyd3hYYk96VWVEZDRESVBydittVzV
wb0FXcjZJdHNWK0gyVlErWkwva0Jud1dIalNUT2FHRmlzcVhZL2FILzFQdEJYQSsxNStZSVdlbUpCU3Yza0RhR
npPWEFFdFI5Wkk4bFlPSmFyblk3QXkvYU42Yjl1R2ZmcmJvL2hWQWNMNFdEZGhrYkJOM204dytnNzZMb0FYTmV
FZXUwNEEvMHhMWnpVQjwvZHM6WDUwOUNlcnRpZmljYXRlPjxkczpYNTA5Q2VydGlmaWNhdGU+TUlJRDZqQ0NBd
EtnQXdJQkFnSUJCREFOQmdrcWhraUc5dzBCQVFzRkFEQk5NUkF3RGdZRFZRUUREQWR5YjI5MExXTmhNUmt3Rnd
ZRFZRUUtEQkJPYjNkcGJtRWdVMjlzZFhScGIyNXpNUkV3RHdZRFZRUUxEQWhRUzBrdFZFVlRWREVMTUFrR0ExV
UVCaE1DVEZVd0hoY05NVFl4TURJMk1EYzFORE13V2hjTk1UZ3dPREkyTURjMU5ETXdXakJOTVJBd0RnWURWUVF
EREFkbmIyOWtMV05oTVJrd0Z3WURWUVFLREJCT2IzZHBibUVnVTI5c2RYUnBiMjV6TVJFd0R3WURWUVFMREFoU
VMwa3RWRVZUVkRFTE1Ba0dBMVVFQmhNQ1RGVXdnZ0VpTUEwR0NTcUdTSWIzRFFFQkFRVUFBNElCRHdBd2dnRUt
Bb0lCQVFDYmJsNXNLQkNqU0I4VE1kYWN5bXgvV2ZPak1XMWdpSWpWSlJZMjhKYk5Xa0NWbXR6Z21pdGdoZnJQU
VBsdWV1MERUYWxiRGtyU1N5aEN2enpQU0dQd0NGT2FoRi9uN2hRYTFGM1VhSFN4VEtyRkM1bk93ZEx6eEtSTzN
XalZ0SUdSU1gya3YxRmZVcFF5NUV5eStyc2ZTeko1OWRVNVpaZFdwZGFEdUR4VlZ5RGVyM0VJeUNiRzcvNUg5T
UQ0WXV6TnBlVEZXbU02Y1VTVDA3OTZYRGxiRXhTVFRHUVhGSkEyK0NDc3k5RFhuSmE4bnowRkU4Zm1jdlFIZVU
2azlRYnB6R2pNZDNEV2xFNm5vN1VkQ1lEMUgwNCt1c0JwNWhoQ3JBQjY3ME5kb1RyTlRtR05BRnQ0SlQwdml0a
ktIcTlLRUlkNkxoZGNtMlRnOTNkRGNXRnRBZ01CQUFHamdkUXdnZEV3RGdZRFZSMFBBUUgvQkFRREFnZUFNRUV
HQTFVZEh3UTZNRGd3TnFBMG9ES0dNR2gwZEhBNkx5OWtjM011Ym05M2FXNWhMbXgxTDNCcmFTMW1ZV04wYjNKN
UwyTnliQzl5YjI5MExXTmhMbU55YkRCTUJnZ3JCZ0VGQlFjQkFRUkFNRDR3UEFZSUt3WUJCUVVITUFLR01HaDB
kSEE2THk5a2MzTXVibTkzYVc1aExteDFMM0JyYVMxbVlXTjBiM0o1TDJOeWRDOXliMjkwTFdOaExtTnlkREFkQ
mdOVkhRNEVGZ1FVKzJ0RnFwTmUzRzIzWVI4eXFCWklpVldTM3ZVd0R3WURWUjBUQVFIL0JBVXdBd0VCL3pBTkJ
na3Foa2lHOXcwQkFRc0ZBQU9DQVFFQUUrTnVkMDVYR09NNkZFWkhXVGM4K2Jtei82QjBYUVhONTY0S1dCQmhja
DlpNUdhZGpxcFNzZXZrbit0ZUxxNW02Q0xvM2U0bFgyZEo3aHNQQXZ5YUxxT0lwemc0OVRHZGliMW5PQjJPNzQ
reUFoVDh2OUZadEgxRUNIWHhZc3V5U0dJS3ZkK0w1SWpKVGlzM2xsNHZVOEZIenlSbE05SVFud1pSNTA2akZjS
mVHbE9neVpoK1VMV25STkVFd3FOOERURjJEMFhvZ1lCc3JDeENCajBQcGFKRnJ1dkVRcXBVdXVZZzE0UjFESkZ
haEx3cVZZU09ENWdQaG1BMEhSNHozR0YzakhTejBpOWpYU1RPc1VjZGt2VUp5MHRJT251anNVRWtnM0g2V2czb
Ho4VHczSWM3VjFOSGIrTUM1SzRadlgrNVNZeXUwK2VyN2JGc2NJclladz09PC9kczpYNTA5Q2VydGlmaWNhdGU
+PC9kczpYNTA5RGF0YT48L2RzOktleUluZm8+PGRzOk9iamVjdD48eGFkZXM6UXVhbGlmeWluZ1Byb3BlcnRpZ
XMgeG1sbnM6eGFkZXM9Imh0dHA6Ly91cmkuZXRzaS5vcmcvMDE5MDMvdjEuMy4yIyIgVGFyZ2V0PSIjaWQtYWZ
kZTc4MjQzNjQ2OGRkNzRlZWIxODFmN2NlMTEwZTEiPjx4YWRlczpTaWduZWRQcm9wZXJ0aWVzIElkPSJ4YWRlc
y1pZC1hZmRlNzgyNDM2NDY4ZGQ3NGVlYjE4MWY3Y2UxMTBlMSI+PHhhZGVzOlNpZ25lZFNpZ25hdHVyZVByb3B
lcnRpZXM+PHhhZGVzOlNpZ25pbmdUaW1lPjIwMTctMDktMjhUMTE6MDk6MDRaPC94YWRlczpTaWduaW5nVGltZ
T48eGFkZXM6U2lnbmluZ0NlcnRpZmljYXRlVjI+PHhhZGVzOkNlcnQ+PHhhZGVzOkNlcnREaWdlc3Q+PGRzOkR
pZ2VzdE1ldGhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyNzaGExIi8+P
GRzOkRpZ2VzdFZhbHVlPmMrVm9oZzBqSWNaNFVRU1dlZ2xDZzBvR05Xcz08L2RzOkRpZ2VzdFZhbHVlPjwveGF
kZXM6Q2VydERpZ2VzdD48eGFkZXM6SXNzdWVyU2VyaWFsVjI+TUZZd1VhUlBNRTB4RURBT0JnTlZCQU1NQjJkd
mIyUXRZMkV4R1RBWEJnTlZCQW9NRUU1dmQybHVZU0JUYjJ4MWRHbHZibk14RVRBUEJnTlZCQXNNQ0ZCTFNTMVV
SVk5VTVFzd0NRWURWUVFHRXdKTVZRSUJDZz09PC94YWRlczpJc3N1ZXJTZXJpYWxWMj48L3hhZGVzOkNlcnQ+P
C94YWRlczpTaWduaW5nQ2VydGlmaWNhdGVWMj48L3hhZGVzOlNpZ25lZFNpZ25hdHVyZVByb3BlcnRpZXM+PHh
hZGVzOlNpZ25lZERhdGFPYmplY3RQcm9wZXJ0aWVzPjx4YWRlczpEYXRhT2JqZWN0Rm9ybWF0IE9iamVjdFJlZ
mVyZW5jZT0iI3ItaWQtMSI+PHhhZGVzOk1pbWVUeXBlPnRleHQveG1sPC94YWRlczpNaW1lVHlwZT48L3hhZGV
zOkRhdGFPYmplY3RGb3JtYXQ+PC94YWRlczpTaWduZWREYXRhT2JqZWN0UHJvcGVydGllcz48L3hhZGVzOlNpZ
25lZFByb3BlcnRpZXM+PHhhZGVzOlVuc2lnbmVkUHJvcGVydGllcz48eGFkZXM6VW5zaWduZWRTaWduYXR1cmV
Qcm9wZXJ0aWVzPjx4YWRlczpTaWduYXR1cmVUaW1lU3RhbXAgSWQ9IlRTLWJiOGJkNDA0LTZiZjAtNDc5ZS1hZ
DhiLTg2ZDUyNDgwYjJkYyI+PGRzOkNhbm9uaWNhbGl6YXRpb25NZXRob2QgQWxnb3JpdGhtPSJodHRwOi8vd3d
3LnczLm9yZy8yMDAxLzEwL3htbC1leGMtYzE0biMiLz48eGFkZXM6RW5jYXBzdWxhdGVkVGltZVN0YW1wIElkP
SJFVFMtYmI4YmQ0MDQtNmJmMC00NzllLWFkOGItODZkNTI0ODBiMmRjIj5NSUlHY0FZSktvWklodmNOQVFjQ29
JSUdZVENDQmwwQ0FRTXhEekFOQmdsZ2hrZ0JaUU1FQWdFRkFEQnlCZ3NxaGtpRzl3MEJDUkFCQktCakJHRXdYd
0lCQVFZREEtnTUVNREV3RFFZSllJWklBV1VEQkFJQkJRQUVJTHB0RmJ0YnRCOWlpSW1SSE4wZXJOZHdxK1hlM0h
1a2sxVnFEeHI4Qll1M0FoRUE5UEpHUTREZzVCLzVPc1pWbkY0T3V4Z1BNakF4T1RBNE1qTXdOakE0TkRSYW9JS
```

```
URjakNDQTI0d2dnSldvQU1DQVFJQ0FXUXdEUVlKS29aSWh2Y05BUUVMQlFBd1ZURVlNQllHQTFVRUF3d1BjMlZ
zWmkxemFXZHVaV1F0ZEhOaE1Sa3dGd1lEVlFRS0RCQk9iM2RwYm1FZ1UyOXNkWFJwYjI1ek1SRXdEd1lEVlFRT
ERBaFFTMGt0VkVWVFZERUxNQWtHQTFVRUJoTUNURlV3SGhjTk1UZ3dPVEl3TURjeE16STJXaGNOTWpBd056SXd
NRGN4TXpJMldqQlZNUmd3RmdZRFZRUUREQTl6Wld4bUxYTnBaMjVsWkMxMGMyRXhHVEFYQmdOVkJBb01FRTV2Z
DJsdVlTQlRiMngxZEdsdmJuTXhFVEFQQmdOVkJBc01DRkJMU1MxVVJWTlVNUXN3Q1FZRFZRUUdFd0pNVlRDQ0F
TSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBTUFaRllYd2ZxQUkvRE9FdkYzankvK1U1L
zNlSjVFRkVQRWIrL24xeStiVkJ5dE0xeHBmbEMyUXdEUEZFKzdzWUgxVkxRZXYwTFlpSmlmNmp5NEQ4WjRSMTV
wc3BhSmNWWkpHN2o2ODRHM2lDcWpVNm9KT05hUktmVmVaK0o4amlGVmIyUnhhSmpyY2pzUW9rbHVCVnllTFVxW
G92WTBDTDJGMWdqRVpKd29nQ0dwUndWWUlab0RlUkVmaS9oWmlMR2xNbnV6SWRQNkpUUUZlK3Nabk5zUkZtOHJ
aSHB1V1BaQkVIL1RzV2pyUGhoOU1leTRablFjWWJ0U2xIelRrdCtBK05WRjRhK1ZESjllcWo1OXNUa2ZKdUs3M
XZ6TUUybHlLcWxldHBhTEtKcmx3NDVobXZrSkloSE1PL0c5Mno5SDBXdVhaQWtMQkg1amQrNjJ2SWxVQ0F3RUF
BYU5KTUVjd0RnWURWUjBQQVFIL0JBUURBZ2VBTUJZR0ExVWRKUUVCL3dRTU1Bb0dDQ3NHQVFVRkJ3TUlNQjBHQ
TFVZERnUVdCQlJuZ2NkaVl6ZWhJbDBFMUlZYkgwdWJOTXZ6bGpBTkJna3Foa2lHOXcwQkFRc0ZBQU9DQVFFQWp
IRnpjY3A1NzRBOFM1RThWdXFWQjhPSzFWaGovb21JbENNM2ZqK2FGOFMzdGZPa1BUZWFFVTVsemFLMGRZditic
XNOUUlpVHZBdzhaUjZWbVdvMDJ6Mks0R29uU3VKWXNJeHlSc2swN1JSZFRscC9VdEpFc0YxWlNXUjl5NnhMS1Y
0KzZpK1hVNnMzSEJmVmwxSURPNXZJSE5NbnoybmhqY0VnM0VjZHNSaVNCc01pVnZYWTZ0My8rbE1TWW9mUThQR
2hzNjg1cTRJTWU2RlZKemtZRTZITFBMME5PYUVQSERaTk9PRDF5TzVRNHdQaGhnY0d4Z0J3emRQVjI0ZW8zYnJ
uWndGOFUyaTdJVElReTVxOVdtR1MvWEJMaExycS9Ybko1OTcrZHY3Q21CM0crd3luZTFZdk5DSzgvcW5ZOVlKd
DJxaXY4dkFQNk9GY1REbHpBekdDQWxzd2dnSlhBZ0VCTUZvd1ZURVlNQllHQTFVRUF3d1BjMlZzWmkxemFXZHV
aV1F0ZEhOaE1Sa3dGd1lEVlFRS0RCQk9iM2RwYm1FZ1UyOXNkWFJwYjI1ek1SRXdEd1lEVlFRTERBaFFTMGt0V
kVWVFZERUxNQWtHQTFVRUJoTUNURlVDQVdRd0RRWUpZSVpJQVdVREJBSUJCUUNnZ2RNd0dnWUpLb1pJaHZjTkF
Ra0RNUTBHQ3lxR1NJYjNEUUVKRUFFRU1Cd0dDU3FHU0liM0RRRUpCVEVQRncweE9UQTRNak13TmpBNE5EUmFNQ
zBHQ1NxR1NJYjNEUUVKTkRFZ01CNHdEUVlKWUlaSUFXVURCQUlCQlFDaERRWUpLb1pJaHZjTkFRRUxCUUF3THd
ZSktvWklodmNOQVFrRU1TSUVJSG5oR1I2MlZYRGlCVElqdlRRTUZvNlBDMndBc0lXVVNXSCs1bjZ0SDE3d01EY
0dDeXFHU0liM0RRRUpFQUl2TVNnd0pqQWtNQ0lFSUdZMDRYdFpleVA5U0RaRTFYMllUTHg5NjRmajNYK1hRSE5
yd1RiWk1sVnRNQTBHQ1NxR1NJYjNEUUVCQ3dVQUJJSUJBQkVRT0xmSi9VdkVsRTNJYmwyc0ppZUNOemNWZVF4R
U1FNFlIZDYwVEtFOWJrb0VWRndpcUNDNW1jMTJNaWx0czFZbGl0RCs5a1VtclhTYk82aytsemZNOFRReGdkRGN
lbmo4aUZTei9xcGdMUURIdm9aOVk1TW9qRWJNeEVYSzMwWVg2SGg3ZVFwbGN5b1VUeTNlUkUrOUxjdHhSV1MyU
XVyWEpTNThPV3N2WWNYVmUwYUJ0OEUwNFppWlp1ZGh4Y1p4ZG1yVThDclNMQThUU29FOTRYaGJ5eVBvcXJJZDN
XZjBhbHpoRHVWR0MwckNoQW02Mk00NWY1R3RHRUF1YkNYcmQzaHRTcGN2eUtLRXF5UFRDdUxpckFnNlhVRnN1R
W5Ld2ZReUxkbXE2UkNQdkFQbGd0NXAvZllCRUl4aEo2QmJQbGx3bkNnU1N0bHI0SkNPdVl3PTwveGFkZXM6RW5
jYXBzdWxhdGVkVGltZVN0YW1wPjwveGFkZXM6U2lnbmF0dXJlVGltZVN0YW1wPjwveGFkZXM6VW5zaWduZWRTa
WduYXR1cmVQcm9wZXJ0aWVzPjwveGFkZXM6VW5zaWduZWRQcm9wZXJ0aWVzPjwveGFkZXM6UXVhbGlmeWluZ1B
yb3BlcnRpZXM+PC9kczpPYmplY3Q+PC9kczpTaWduYXR1cmU+",
  "digestAlgorithm" : null,
  "name" : "xades-detached-extended-xades-baseline-t.xml"
}
```


Timestamp document

The method allows timestamping of a provided document. Available for PDF, ASiC-E and ASiC-S container formats.

Request

```
POST /services/rest/signature/one-document/signDocument HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 2842
```

```
{
  "timestampParameters" : {
    "digestAlgorithm" : "SHA512",
    "canonicalizationMethod" : null,
    "timestampContainerForm" : "PDF"
  },
  "toTimestampDocument" : {
    "bytes" :
"JVBERi0xLjYNJeLjz9MNCjExIDAgb2JqDTw8L0xpbmVhcml6ZWQgMS9MIDQ4ODAvTyAxMy9FIDc2NC9OIDEvV
CA0NTc3L0ggWyA0MzEgMTMxXT4+DWVuZG9iag0gICAgICAgICAgICAgICAgICAgICAgDQoxNCAwIG9iag08PC9
EZWNvZGVQYXJtczw8L0NvbHVtbnMgMy9QcmVkaWN0b3IgMTI+Pi9GaWx0ZXIvRmxhdGVEZWNvZGUvSURbPDVGR
UVEN0UwNjFCRTc0NDA4MTI5OEE3M0UwREM3NzZCPjw3OEI2REQzNUIyRThFMjQ0ODJENzhCQ0Y5NzQ2MTc4OD5
dL0luZGV4WzExIDVdL0luZm8gMTAgMCBSL0xlbmd0aCAzMC9QcmV2IDQ1NzgvUm9vdCAxMiAwIFIvU2l6ZSAxN
i9UeXBlL1hSZWYvV1sxIDIgMF0+PnN0cmVhbQ0KaN5iYmQQYGJgUmJiYPBhYvj3jYmB0RogwAARMAK8DQplbmR
zdHJlYW0NZW5kb2JqDXN0YXJ0eHJlZg0KMA0KJSVFT0YNCiAgICAgICAgDQoxNSAwIG9iag08PC9GaWx0ZXIvR
mxhdGVEZWNvZGUvSSA3Ny9MZW5ndGggNDcvTyA2MS9TIDM2Pj5zdHJlYW0NCmjeYmBgYASi3wwgUMeACVgYOJB
4jFAMVwvkMP2BCt4DK2cQlIbyjwMEGAC77QTeDQplbmRzdHJlYW0NZW5kb2JqDTEyIDAgb2JqDTw8L01ldGFkY
XRhIDIgMCBSL091dGxpbmVzIDYgMCBSL1BhZ2VzIDkgMCBSL1R5cGUvQ2F0YWxvZz4+DWVuZG9iag0xMyAwIG9
iag08PC9Dcm9wQm94WzAuMCAwLjAgNjEyLjAgNzkyLjBdL01lZGlhQm94WzAuMCAwLjAgNjEyLjAgNzkyLjBdL
1BhcmVudCA5IDAgUi9SZXNvdXJjZXM8PD4+L1JvdGF0ZSAwL1R5cGUvUGFnZT4+DWVuZG9iag0xIDAgb2JqDTw
8L0ZpbHRlci9GbGF0ZURlY29kZS9GaXJzdCAxNC9MZW5ndGggMTI4L04gMy9UeXBlL09ialN0bT4+c3RyZWFtD
Qpo3jJTMFAwVzCxVLBQsDRWsLHRd84vzStRMNR3yywqLgFKGSgE6fskwpkhlQWp+v6lJTmZeanFdnZADY5ArSC
ZgMSiVKBOM4iyzJKcVA2f1NSiVIXg1MySVE2wWpdoQ2OwfERkFMhiSyOFvNKcnFj9YH33/JB8OzuAAAMA2N0nu
Q0KZW5kc3RyZWFtDWVuZG9iag0yIDAgb2JqDTw8L0xlbmd0aCAzMTc1L1N1YnR5cGUvWE1ML1R5cGUvTWV0YWR
hdGE+PnN0cmVhbQ0KPD94cGFja2V0IGJlZ2luPSLvu78iIGlkPSJXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQiP
z4KPHg6eG1wbWV0YSB4bWxuczp4PSJhZG9iZTpuczptZXRhLyIgeDp4bXB0az0iQWRvYmUgWE1QIENvcmUgNS4
0LWMwMDYgODAuMTU5ODI1LCAyMDE2LzA5LzE2LTAzOjMxOjA4ICAgICAgICAiPgogICA8cmRmOlJERiB4bWxuc
zpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPgogICAgICA8cmRmOkR
lc2NyaXB0aW9uIHJkZjphYm91dD0iIgogICAgICAgICAgICB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZG9iZS5jb
20veGFwLzEuMC8iCiAgICAgICAgICAgIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4
xLyIKICAgICAgICAgICAgeG1sbnM6eG1wTU09Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9tbS8iCiAgI
CAgICAgICAgIHhtbG5zOnBkZj0iaHR0cDovL25zLmFkb2JlLmNvbS9wZGYvMS4zLyI+CiAgICAgICAgIDx4bXA
6TW9kaWZ5RGF0ZT4yMDE4LTAzLTA5VDEyOjAxOjU0KzAxOjAwPC94bXA6TW9kaWZ5RGF0ZT4KICAgICAgICAgP
HhtcDpDcmVhdGVEYXRlPjIwMTgtMDMtMDlUMTI6MDE6MzErMDE6MDA8L3htcDpDcmVhdGVEYXRlPgogICAgICA
gICA8eG1wOk1ldGFkYXRhRGF0ZT4yMDE4LTAzLTA5VDEyOjAxOjU0KzAxOjAwPC94bXA6TWV0YWRhdGFEYXRlP
gogICAgICAgICA8eG1wOkNyZWF0b3JUb29sPkFkb2JlIEFjcm9iYXQgUHJvIDExLjAuMjM8L3htcDpDcmVhdG9
yVG9vbD4KICAgICAgICAgPGRjOmZvcm1hdD5hcHBsaWNhdGlvbi9wZGY8L2RjOmZvcm1hdD4KICAgICAgICAgP
HhtcE1NOkRvY3VtZW50SUQ+dXVpZDo5NWZjN2FhNi02NjgxLTQ1M2EtODNiZi1hYmExYTM2ZTM1OGY8L3htcE1
NOkRvY3VtZW50SUQ+CiAgICAgICAgIDx4bXBNTTpJbnN0YW5jZUlEPnV1aWQ6OTZjZDE0ZjQtYjVmYS00ODUxL
Tg1ZmUtZGI5ZGY0ZDIyZTgzPC94bXBNTTpJbnN0YW5jZUlEPgogICAgICAgICA8cGRmOlByb2R1Y2VyPkFkb2J
lIEFjcm9iYXQgUHJvIDExLjAuMjM8L3BkZjpQcm9kdWNlcj4KICAgICAgPC9yZGY6RGVzY3JpcHRpb24+CiAgI
DwvcmRmOlJERj4KPC94OnhtcG1ldGE+CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
```

```
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAo
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgI
CAgICAgICAKPD94cGFja2V0IGVuZD0idyI/Pg0KZW5kc3RyZWFtDWVuZG9iag0zIDAgb2JqDTw8L0ZpbHRlci9
GbGF0ZURlY29kZS9GaXJzdCA0L0xlbmd0aCA0OS9OIDEvVHlwZS9PYmpTdG0+PnN0cmVhbQ0KaN6yVDBQsLHRd
84vzStRMNT3zkwpjjY0BgoGxeqHVBak6gckpqcW29kBBBgA4DULrw0KZW5kc3RyZWFtDWVuZG9iag00IDAgb2J
qDTw8L0ZpbHRlci9GbGF0ZURlY29kZS9GaXJzdCA1L0xlbmd0aCAxMDYvTiAxL1R5cGUvT2JqU3RtPj5zdHJlY
W0NCmjeMjRQMFCwsdF3LkpNLMnMz3NJLEnVcLEyMjC0MDA2sDQEMowNtQ0M1Q0M1DUhqvKLNBxT8pNSFRyTi/K
TEksUAoryFQwN9Qz0jIw19X3zU7CYYWoCNwOoOqU0ORWfIXZ2AAEGAByrKRQNCmVuZHN0cmVhbQ1lbmRvYmoNN
SAwIG9iag08PC9EZWNvZGVQYXJtczw8L0NvbHVtbnMgNC9QcmVkaWN0b3IgMTI+Pi9GaWx0ZXIvRmxhdGVEZWN
vZGUvSURbPDVGRUVEN0UwNjFCRTc0NDA4MTI5OEE3M0UwREM3NzZCPjw3OEI2REQzNUIyRThFMjQ0ODJENzhCQ
0Y5NzQ2MTc4OD5dL0luZm8gMTAgMCBSL0xlbmd0aCA0OC9Sb290IDEyIDAgUi9TaXplIDExL1R5cGUvWFJlZi9
XWzEgMiAxXT4+c3RyZWFtDQpo3mJiAAImRqY/DEwMjPeABO8WEKsXSDAcB0q8lwexGBhhBNM/IMHIABBgALdBB
h0NCmVuZHN0cmVhbQ1lbmRvYmoNc3RhcnR4cmVmDQoxMTYNCiUlRU9GDQo=",
    "digestAlgorithm" : null,
    "name" : "sample.pdf"
  }
}
```


Response

```
HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 06:08:46 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
```

```
Transfer-Encoding: chunked
Content-Length: 1769

{
  "bytes":
"JVBERi0xLjYNJeLjz9MNCjExIDAgb2JqDTw8L0xpbmVhcml6ZWQgMS9MIDQ4ODAvTyAxMy9FIDc2NC9OIDEvV
CA0NTc3L0ggWyA0MzEgMTMxXT4+DWVuZG9iag0gICAgICAgICAgICAgICAgICAgICAgDQoxNCAwIG9iag08PC9
EZWNvZGVQYXJtczw8L0NvbHVtbnMgMy9QcmVkaWN0b3IgMTI+Pi9GaWx0ZXIvRmxhdGVEZWNvZGUvSURbPDVGR
UVEN0UwNjFCRTc0NDA4MTI5OEE3M0UwREM3NzZCPjw3OEI2REQzNUIyRThFMjQ0ODJENzhCQ0Y5NzQ2MTc4OD5
dL0luZGV4WzExIDVdL0luZm8gMTAgMCBSL0xlbmd0aCAzMC9QcmV2IDQ1NzgvUm9vdCAxMiAwIFIvU2l6ZSAxN
i9UeXBlL1hSZWYvV1sxIDIgMF0+PnN0cmVhbQ0KaN5iYmQQYGJgUmJiYPBhYvj3jYmB0RogwAARMAK8DQplbmR
zdHJlYW0NZW5kb2JqDXN0YXJ0eHJlZg0KMA0KJSVFT0YNCiAgICAgICAgDQoxNSAwIG9iag08PC9GaWx0ZXIvR
mxhdGVEZWNvZGUvSSA3Ny9MZW5ndGggNDcvTyA2MS9TIDM2Pj5zdHJlYW0NCmjeYmBgYASi3wwgUMeACVgYOJB
4jFAMVwvkMP2BCt4DK2cQlIbyjwMEGAC77QTeDQplbmRzdHJlYW0NZW5kb2JqDTEyIDAgb2JqDTw8L01ldGFkY
XRhIDIgMCBSL091dGxpbmVzIDYgMCBSL1BhZ2VzIDkgMCBSL1R5cGUvQ2F0YWxvZz4+DWVuZG9iag0xMyAwIG9
iag08PC9Dcm9wQm94WzAuMCAwLjAgNjEyLjAgNzkyLjBdL01lZGlhQm94WzAuMCAwLjAgNjEyLjAgNzkyLjBdL
1BhcmVudCA5IDAgUi9SZXNvdXJjZXM8PD4+L1JvdGF0ZSAwL1R5cGUvUGFnZT4+DWVuZG9iag0xIDAgb2JqDTw
8L0ZpbHRlci9GbGF0ZURlY29kZS9GaXJzdCAxNC9MZW5ndGggMTI4L04gMy9UeXBlL09ialN0bT4+c3RyZWFtD
Qpo3jJTMFAwVzCxVLBQsDRWsLHRd84vzStRMNR3yywqLgFKGSgE6fskwpkhlQWp+v6lJTmZeanFdnZADY5ArSC
ZgMSiVKBOM4iyzJKcVA2f1NSiVIXg1MySVE2wWpdoQ2OwfERkFMhiSyOFvNKcnFj9YH33/JB8OzuAAAMA2N0nu
Q0KZW5kc3RyZWFtDWVuZG9iag0yIDAgb2JqDTw8L0xlbmd0aCAzMTc1L1N1YnR5cGUvWE1ML1R5cGUvTWV0YWR
hdGE+PnN0cmVhbQ0KPD94cGFja2V0IGJlZ2luPSLvu78iIGlkPSJXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQiP
z4KPHg6eG1wbWV0YSB4bWxuczp4PSJhZG9iZTpuczptZXRhLyIgeDp4bXB0az0iQWRvYmUgWE1QIENvcmUgNS4
0LWMwMDYgODAuMTU5ODI1LCAyMDE2LzA5LzE2LTAzOjMxOjA4ICAgICAgICAiPgogICA8cmRmOlJERiB4bWxuc
zpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPgogICAgICA8cmRmOkR
lc2NyaXB0aW9uIHJkZjphYm91dD0iIgogICAgICAgICAgICB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZG9iZS5jb
20veGFwLzEuMC8iCiAgICAgICAgICAgIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4
xLyIKICAgICAgICAgICAgeG1sbnM6eG1wTU09Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9tbS8iCiAgI
CAgICAgICAgIHhtbG5zOnBkZj0iaHR0cDovL25zLmFkb2JlLmNvbS9wZGYvMS4zLyI+CiAgICAgICAgIDx4bXA
6TW9kaWZ5RGF0ZT4yMDE4LTAzLTA5VDEyOjAxOjU0KzAxOjAwPC94bXA6TW9kaWZ5RGF0ZT4KICAgICAgICAgP
HhtcDpDcmVhdGVEYXRlPjIwMTgtMDMtMDlUMTI6MDE6MzErMDE6MDA8L3htcDpDcmVhdGVEYXRlPgogICAgICA
gICA8eG1wOk1ldGFkYXRhRGF0ZT4yMDE4LTAzLTA5VDEyOjAxOjU0KzAxOjAwPC94bXA6TWV0YWRhdGFEYXRlP
gogICAgICAgICA8eG1wOkNyZWF0b3JUb29sPkFkb2JlIEFjcm9iYXQgUHJvIDExLjAuMjM8L3htcDpDcmVhdG9
yVG9vbD4KICAgICAgICAgPGRjOmZvcm1hdD5hcHBsaWNhdGlvbi9wZGY8L2RjOmZvcm1hdD4KICAgICAgICAgP
HhtcE1NOkRvY3VtZW50SUQ+dXVpZDo5NWZjN2FhNi02NjgxLTQ1M2EtODNiZi1hYmExYTM2ZTM1OGY8L3htcE1
NOkRvY3VtZW50SUQ+CiAgICAgICAgIDx4bXBNTTpJbnN0YW5jZUlEPnV1aWQ6OTZjZDE0ZjQtYjVmYS00ODUxL
Tg1ZmUtZGI5ZGY0ZDIyZTgzPC94bXBNTTpJbnN0YW5jZUlEPgogICAgICAgICA8cGRmOlByb2R1Y2VyPkFkb2J
lIEFjcm9iYXQgUHJvIDExLjAuMjM8L3BkZjpQcm9kdWNlcj4KICAgICAgPC9yZGY6RGVzY3JpcHRpb24+CiAgI
DwvcmRmOlJERj4KPC94OnhtcG1ldGE+CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICA
```

```
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICiAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAo
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICiAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAKPD94cGFja2V0IGVuZD0idyI/Pg0KZW5kc3RyZWFtDWVuZG9iag0zIDAgb2JqDTw8L0ZpbHRlci9
GbGF0ZURlY29kZS9GaXJzdCA0L0xlbmd0aCA0OS9OIDEvVHlwZS9PYmpTdG0+PnN0cmVhbQ0KaN6yVDBQsLHRd
84vzStRMNT3zkwpjjY0BgoGxeqHVBak6gckpqcW29kBBBgA4DULrw0KZW5kc3RyZWFtDWVuZG9iag00IDAgb2J
qDTw8L0ZpbHRlci9GbGF0ZURlY29kZS9GaXJzdCA1L0xlbmd0aCAxMDYvTiAxL1R5cGUvT2JqU3RtPj5zdHJlY
W0NCmjeMjRQMFCwsdF3LkpNLMnMz3NJLEnVcLEyMjC0MDA2sDQEMowNtQ0M1Q0M1DUhqvKLNBxT8pNSFRyTi/K
TEksUAoryFQwN9Qz0jIw19X3zU7CYYWoCNwOoOqU0ORWfIXZ2AAEGAByrKRQNCmVuZHN0cmVhbQ1lbmRvYmoNN
SAwIG9iag08PC9EZWNvZGVQYXJtczw8L0NvbHVtbnMgNC9QcmVkaWN0b3IgMTI+Pi9GaWx0ZXIvRmxhdGVEZWN
vZGUvSURbPDVGRUVEN0UwNjFCRTc0NDA4MTI5OEE3M0UwREM3NzZCPjw3OEI2REQzNUIyRThFMjQ0ODJENzhCQ
0Y5NzQ2MTc4OD5dL0luZm8gMTAgMCBSL0xlbmd0aCA0OC9Sb290IDEyIDAgUi9TaXplIDExL1R5cGUvWFJlZi9
XWzEgMiAxXT4+c3RyZWFtDQpo3mJiAAImRqY/DEwMjPeABO8WEKsXSDAcB0q8lwexGBhhBNM/IMHIABBgALdBB
h0NCmVuZHN0cmVhbQ1lbmRvYmoNc3RhcnR4cmVmDQoxMTYNCiUlRU9GDQoNCjEyIDAgb2JqCjw8Ci9NZXRhZGF
0YSAyIDAgUgovT3V0bGluZXMgNiAwIFIKL1BhZ2VzIDkgMCBSCi9UeXBlIC9DYXRhbG9nCi9BY3JvRm9ybSA8P
AovRmllbGRzIFsxNiAwIFJdCi9TaWdGbGFncyAzCj4+Cj4+CmVuZG9iagoxNiAwIG9iago8PAovRlQgL1NpZwo
vVHlwZSAvQW5ub3QKL1N1YnR5cGUgL1dpZGdldAovRiAxMzIKL1QgKFNpZ25hdHVyZTEpCi9WIDE3IDAgUgovU
CAxMyAwIFIKL1JlY3QgWzAuMCAwLjAgMC4wIDAuMF0KL0FQIDw8Ci9OIDE4IDAgUgo+Pgo+PgplbmRvYmoKMTc
gMCBvYmoKPDwKL1R5cGUgL0RvY1RpbWVTdGFtcAovRmlsdGVyIC9BZG9iZS5QUEtMaXRlCi9TdWJGaWx0ZXIgL
0VUU0kuUkZDMzE2MQovQ29udGVudHMgPDMwODIwNkUxMDYwOTJBODY0ODg2RjcwRDAxMDcwMkEwODIwNkQyMzA
4MjA2Q0UwMjAxMDMzMTBGMzAwRDA2MDk2MDg2NDgwMTY1MDMwNDAyMDMwNTAwMzA4MTk0MDYwQjJBODY0ODg2R
jcwRDAxMDkxMDAxMDRBMDgxODQwNDgxODEzMDdGMDIwMTAxMDYwMzJBMDMwNDMwNTEzMDBEMDYwOTYwODY0ODA
xNjUwMzA0MDIwMzA1MDAwNDQwMEMyRDA4MjA1MEIwQzY3QTlCNUVGMzcyRDFEREJBMTg4NDVFRDJFNzJFNkQzR
EFFMEQzMTdCQjUwN0Y1Mjg2NUJEMjZENTcyMDM3NjYyQjY1M0VCRkI4N0FFNkVBRjQwMzk2QkMwRDU1M0ZGRDM
2RkNEMEJEQjU2NjkzRTNBNEQwMjExMDBGNEE0NjgyNkFBNTgxQzkwOEJGNzMxN0ZFMzY1RTMwQzE4MEYzMjMwM
zIzMDMwMzEzMTM1MzAzNzM0MzAzMTM4NUFBMDgyMDM3MjMwODIwMzZFMzA4MjAyNTZBMDAzMDIwMTAyMDIwMTY
0MzAwRDA2MDkyQTg2NDg4NkY3MEQwMTAxMEIwNTAwMzA1NTMxMTgzMDE2MDYwMzU1MDQwMzBDMEY3MzY1NkM2N
jJENzM2OTY3NkU2NTY0MkQ3NDczNjEzMTE5MzAxNzA2MDM1NTA0MEEwQzEwNEU2Rjc3Njk2RTYxMjA1MzZGNkM
3NTc0Njk2RjZFNzMzMTExMzAwRjA2MDM1NTA0MEIwQzA4NTA0QjQ5MkQ1NDQ1NTM1NDMxMEIzMDA5MDYwMzU1M
DQwNjEzMDI0QzU1MzAxRTE3MEQzMTM4MzAzOTMyMzAzMDM3MzEzMzMyMzY1QTE3MEQzMjMwMzAzNzMyMzAzMDM
3MzEzMzMyMzY1QTMwNTUzMTE4MzAxNjA2MDM1NTA0MDMwQzBGNzM2NTZDNjYyRDczNjk2NzZFNjU2NDJENzQ3M
zYxMzExOTMwMTcwNjAzNTUwNDBBMEMxMDRFNkY3NzY5NkU2MTIwNTM2RjZDNzU3NDY5NkY2RTczMzExMTMwMEY
wNjAzNTUwNDBCMEMwODUwNEI0OTJENTQ0NTUzNTQzMTBCMzAwOTA2MDM1NTA0MDYxMzAyNEM1NTMwODIwMTIyM
zAwRDA2MDkyQTg2NDg4NkY3MEQwMTAxMDEwNTAwMDM4MjAxMEYwMDMwODIwMTBBMDI4MjAxMDEwMEMwMTkxNTg
1RjA3RUEwMDhGQzMzODRCQzVERTNDQkZGOTRFN0ZEREUyNzkxMDUxMEYxMUJGQkY5RjVDQkU2RDUwNzJCNENEN
```

```
zFBNUY5NDJEOTBDMDMzQzUxM0VFRUM2MDdENTUyRDA3QUZEMEI2MjIyNjI3RkE4RjJFMDNGMTlFMTFENzlBNkN
BNUEyNUM1NTkyNDZFRTNFQkNFMDZERTIwQUE4RDRFQTgyNEUzNUE0NEE3RDU3OTlGODlGMjM4ODU1NUJEOTFDN
UEyNjNBREM4RUM0Mjg5MjVCODE1NzI3OEI1MkE1RThCRDhEMDIyRjYxNzU4MjMxMTkyNzBBMjAwODZBNTFDMTU
2MDg2NjgwREU0NDQ3RTJGRTE2NjIyQzY5NEM5RUVDQzg3NEZFODk0RDAxNUVGQUM2NjczNkM0NDU5QkNBRDkxR
TlCOTYzRDkwNDQxRkY0RUM1QTNBQ0Y4NjFGNEM3QjJFMTk5RDA3MTg2RUQ0QTUxRjM0RTRCN0UwM0UzNTUxNzg
2QkU1NDMyN0Q3QUE4RjlGNkM0RTQ3QzlCOEFFRjVCRjMzMDREQTVDOEFBQTU3QURBNUEyQ0EyNkI5NzBFMzk4N
jZCRTQyNDg4NDczMEVGQzZGNzZDRkQxRjQ1QUU1RDkwMjQyQzExRjk4RERGQkFEQUYyMjU1MDIwMzAxMDAwMUE
zNDkzMDQ3MzAwRTA2MDM1NTFEMEYwMTAxRkYwNDA0MDMwMjA3ODAzMDE2MDYwMzU1MUQyNTAxMDFGRjA0MEMzM
DBBMDYwODJCMDYwMTA1MDUwNzAzMDgzMDFEMDYwMzU1MUQwRTA0MTYwNDE0Njc4MUM3NjI2MzM3QTEyMjVEMDR
ENDg2MUIxRjRCOUIzNENCRjM5NjMwMEQwNjA5MkE4NjQ4ODZGNzBEMDEwMTBCMDUwMDAzODIwMTAxMDA4QzcxN
zM3MUNBNzlFRjgwM0M0QjkxM0M1NkVBOTUwN0MzOEFENTU4NjNGRTg5ODg5NDIzMzc3RTNGOUExN0M0QjdCNUY
zQTQzRDM3OUExMTRFNjVDREEyQjQ3NThCRkU2RUFCMEQ0MDg4OTNCQzBDM0M2NTFFOTU5OTZBMzREQjNEOEFFM
DZBMjc0QUUyNThCMDhDNzI0NkM5MzRFRDE0NUQ0RTVBN0Y1MkQyNDRCMDVENTk0OTY0N0RDQkFDNEIyOTVFM0V
FQTJGOTc1M0FCMzcxQzE3RDU5NzUyMDMzQjlCQzgxQ0QzMjdDRjY5RTE4REMxMjBEQzQ3MURCMTE4OTIwNkMzM
jI1NkY1RDhFQURERkZGQTUzMTI2Mjg3RDBGMEYxQTFCM0FGMzlBQjgyMEM3QkExNTUyNzM5MTgxM0ExQ0IzQ0J
EMEQzOUExMEYxQzM2NEQzOEUwRjVDOEVFNTBFMzAzRTE4NjA3MDZDNjAwNzBDREQzRDVEQjg3QThEREJBRTc2N
zAxN0M1MzY4QkIyMTMyMTBDQjlBQkQ1QTYxOTJGRDcwNEI4NEJBRUFGRDc5QzlFN0RFRkU3NkZFQzI5ODFEQzZ
GQjBDQTc3QjU2MkYzNDIyQkNGRUE5RDhGNTgyNkREQUE4QUZGMkYwMEZFOEUxNUM0QzM5NzMwMzMxODIwMkE5M
zA4MjAyQTUwMjAxMDEzMDVBMzA1NTMxMTgzMDE2MDYwMzU1MDQwMzBDMEY3MzY1NkM2NjJENzM2OTY3NkU2NTY
0MkQ3NDczNjEzMTE5MzAxNzA2MDM1NTA0MEEwQzEwNEU2Rjc3Njk2RTYxMjA1MzZGNkM3NTc0Njk2RjZFNzMzM
TExMzAwRjA2MDM1NTA0MEIwQzA4NTA0QjQ5MkQ1NDQ1NTM1NDMxMEIzMDA5MDYwMzU1MDQwNjEzMDI0QzU1MDI
wMTY0MzAwRDA2MDk2MDg2NDgwMTY1MDMwNDAyMDMwNTAwQTA4MjAxMjAzMDFBMDYwOTJBODY0ODg2RjcwRDAxM
DkwMzMxMEQwNjBCMkE4NjQ4ODZGNzBEMDEwOTEwMDEwNDMwMUMwNjA5MkE4NjQ4ODZGNzBEMDEwOTA1MzEwRjE
3MEQzMjMwMzAzMTMxMzUzMDM3MzQzMDMxMzg1QTMwMkQwNjA5MkE4NjQ4ODZGNzBEMDEwOTM0MzEyMDMwMUUzM
DBEMDYwOTYwODY0ODAxNjUwMzA0MDIwMzA1MDBBMTBEMDYwOTJBODY0ODg2RjcwRDAxMDEwRDA1MDAzMDRGMDY
wOTJBODY0ODg2RjcwRDAxMDkwNDMxNDIwNDQwNTc1MTFCNTAzRTYxN0Y5OTJBRTg3OUUwMDBBRkYyMkVFOUYxN
jk4RkVGN0NCNEI1ODJDODFFQUM2MzU4Mjg2MzE0ODk1NEQ3NTE0RjE4MUQ2NUVEN0EyNTg5NTE0Q0Q5MEYyQzU
0ODZGMjE0REQ4RDYxQjRBRTNDNDcyOEZCNkIzMDY0MDYwQjJBODY0ODg2RjcwRDAxMDkxMDAyMkYzMTU1MzA1M
zMwNTEzMDRGMzAwQjA2MDk2MDg2NDgwMTY1MDMwNDAyMDMwNDQwNkFEQTM3RUZDRThEMTNERTE5ODA5RkI2QjR
EQTMxMEZGRTEwRjM4QUE2RUI4NTgwMUNFQTU3Qzg2MjYyNzc5RTI0NzFENTcwNTk2OEE3RTVDM0M1OTA0OTQ4M
jhGNkU4QzEwMTAzNDg5QzBGQzUyMjQzMjk4NDY1QTgwQzFGQTczMDBEMDYwOTJBODY0ODg2RjcwRDAxMDEwRDA
1MDAwNDgyMDEwMDk2NUYzRENENTU3NkVFM0EwMkNGOTk4QzcxNDE5QjFDQzMzMjMyQTc2OUYwQTg5NEE4RDk1M
kNCRDQ2MzczNTk3M0I4OTE0RUJBODBENkYyREI5QTE2RTg0MjE1QTEzRjE2MzM5QkE3NDQ0QTkxRjRGNjk0QjI
2MjY3N0YxNTFENENFNTc2NkMzNjBCN0EwQTczQzRGMDczQ0Y2MjVFMjc4ODBFMUQxQTQyOTUwRjk1QzIxNDU2N
0QyNTlFOUU4MTdFRkVGOTYxQUQ0ODYyRTZEMjIwMzA3ODRGMThCREE5QTExNEVDMzQ5RTc1NjRDRThCRDU3NkJ
GNzg1Q0IwOTQ2QjM3QTNENjhCQzNBNUU5MDBCQkY5NjlFQzlGM0ZDRDA2RkVBMDI4N0E5MENDQjVDM0Q2MkREO
UM4NDE2RUYzNTVFNTdCMkZGQzUwQTVBODcxNDUyRUYyN0ExQjQ4MTczQUY0MjRDMUIyOTYwMkE5QkJCOTA5REM
0MjQzQzJFRUNEODNBNkZEQjgwMkFDODc5RkYwQzQ5MzhFQzZCNjRBN0E2REUxMTQ3RUY5OTJCM0Q2NjQ2RjkwQ
TJFREYxRTJENzdFRTIxNjdBNDc4MzdCRjEzNzc2MjE0NEQ3OUQ3N0JDNjA0NzhFNUUxRkU4MTNCMTE4REI3RUI
yNDc0RU...+PgplbmRvYmoKMTMgMCBvYmoKPDwKL0Nyb3BCb3ggWzAuMCAwLjAgNjEyLjAgNzkyLjBdCi9NZWR
pYUJveCBbMC4wIDAuMCA2MTIuMCA3OTIuMF0KL1BhcmVudCA5IDAgUgovUmVzb3VyY2VzIDw8Cj4+Ci9Sb3Rhd
GUgMAovVHlwZSAvUGFnZQovQW5ub3RzIFsxNiAwIFJdCj4+CmVuZG9iagoxOCAwIG9iago8PAovTGVuZ3RoIDA
KL1R5cGUgL1hPYmplY3QKL1N1YnR5cGUgL0Zvcm0KL0JCb3ggWzAuMCAwLjAgMC4wIDAuMF0KPj4Kc3RyZWFtD
QoNCmVuZHN0cmVhbQplbmRvYmoKMTkgMCBvYmoKPDwKL0xlbmd0aCAyNgovSUQgWzw1RkVFRDdFMDYxQkU3NDQ
wODEyOThBNzNFMERDNzc2Qj4gPERBQ0NGNUZDNzNDM0U1ODIxM0Y1N0Q1MkUyQzJBRkJEPl0KL0luZm8gMTAgM
CBSCi9Sb290IDEyIDAgUgovUHJldiAxMTYKL1R5cGUgL1hSZWYKL1NpemUgNTMKL0luZGV4IFswIDEgMTIgMiA
xNiAzXQovVyBbMSAyIDBdCi9GaWx0ZXIgL0ZsYXRlRGVjb2RlCj4+CnN0cmVhbQ0KeJxjYGBgFBZijNvDKDyRU
USVMT4cABQxAtgNCmVuZHN0cmVhbQplbmRvYmoKc3RhcnR4cmVmCjI0NTEzCiUlRU9GCg==",
  "digestAlgorithm": null,
  "name": "sample-timestamped.pdf"
}
```

REST server signature service

This service also exposed 4 methods :

Rest server signing service

```
// Instantiate a RestSignatureTokenConnection
RestSignatureTokenConnection remoteToken = new RestSignatureTokenConnectionImpl();

// Retrieves available keys on server side
List<RemoteKeyEntry> keys = remoteToken.getKeys();

String alias = keys.get(0).getAlias();

// Retrieves a key on the server side by its alias
RemoteKeyEntry key = remoteToken.getKey(alias);

DSSDocument documentToSign = new InMemoryDocument("Hello world!".getBytes());

// Create a toBeSigned DTO
ToBeSignedDTO toBeSigned = new ToBeSignedDTO(DSSUtils.toByteArray(documentToSign));

// Signs the document with a given Digest Algorithm and alias for a key to use
// Signs the digest value with the given key
SignatureValueDTO signatureValue = remoteToken.sign(toBeSigned, DigestAlgorithm.
SHA256, alias);

// Or alternatively we can sign the document by providing digest only

// Prepare digestDTO.
// NOTE: the used Digest algorithm must be the same!
DigestDTO digestDTO = new DigestDTO(DigestAlgorithm.SHA256, DSSUtils.digest
(DigestAlgorithm.SHA256, documentToSign));

// Signs the digest
SignatureValueDTO signatureValueFromDigest = remoteToken.signDigest(digestDTO, alias);
```

Get keys

This method allows retrieving of all available keys on the server side (PKCS#11, PKCS#12, HSM,...). All keys will have an alias, a signing certificate and its chain. The alias will be used in following steps.

Request

```
GET /services/rest/server-signing/keys HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Host: localhost:8080
```

Response

```
HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 06:08:46 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
Transfer-Encoding: chunked
Content-Length: 2189

[ {
  "alias" : "certificate",
  "encryptionAlgo" : "RSA",
  "certificate" : {
    "encodedCertificate" :
"MIIC6jCCAdKgAwIBAgIGLtYU17tXMA0GCSqGSIb3DQEBCwUAMDAxGzAZBgNVBAMMElJvb3RTZWxmU2lnbmVkR
mFrZTERMA8GA1UECgwIRFNTLXRlc3QwHhcNMTcwNjA4MTEyNjAxWhcNNDcwNzA0MDc1NzI0WjAoMRMwEQYDVQQ
DDApTaWduZXJGYWtlMREwDwYDVQQKDAhEU1MtdGVzdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
MI3kZhtnipn+iiZHZ9ax8FlfE5Ow/cFwBTfAEb3R1ZQUp6/BQnBt7Oo0JWBtc9qkv7JUDdcBJXPV5QWS5AyMPH
pqQ75Hitjsq/Fzu8eHtkKpFizcxGa9BZdkQjh4rSrtO1Kjs0Rd5DQtWSgkeVCCN09kN0ZsZ0ENY+Ip8QxSmyzt
sStkYXdULqpwz4JEXW9vz64eTbde4vQJ6pjHGarJf1gQNEc2XzhmI/prXLysWNqC7lZg7PUZUTrdegABTUzYCR
J1kWBRPm4qo0LN405c94QQd45a5kTgowHzEgLnAQI28x0M3A59TKC+ieNc6VF1PsTLpUw7PNI2VstX5jAuasCA
wEAAaMSMBAwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBCwUAA4IBAQCK6LGA01TR+rmU8p6yhAi4OkDN2b1
dbIL8l8iCMYopLCxx8xqq3ubZCOxqh1X2j6pgWzarb0b/MUix00IoUvNbFOxAW7PBZIKDLnm6LsckRxs1U32sC
9d1LOHe3WKBNB6GZALT1ewjh7hSbWjftlmcovq+6eVGA5cvf2u/2+TkKkyHV/NR394nXrdsdpvygwypEtXjetz
D7UT93Nuw3xcV8VIftIvHf9LjU7h+UjGmKXG9c15eYr3SzUmv6kyOI0Bvw14PWtsWGl0QdOSRvIBBrP4adCnGT
gjgjk9LTcO8B8FKrr+8lHGuc0bp4lIUToiUkGILXsiEeEg9WAqm+XqO"
  },
  "certificateChain" : [ {
    "encodedCertificate" :
"MIIC6jCCAdKgAwIBAgIGLtYU17tXMA0GCSqGSIb3DQEBCwUAMDAxGzAZBgNVBAMMElJvb3RTZWxmU2lnbmVkR
mFrZTERMA8GA1UECgwIRFNTLXRlc3QwHhcNMTcwNjA4MTEyNjAxWhcNNDcwNzA0MDc1NzI0WjAoMRMwEQYDVQQ
DDApTaWduZXJGYWtlMREwDwYDVQQKDAhEU1MtdGVzdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
MI3kZhtnipn+iiZHZ9ax8FlfE5Ow/cFwBTfAEb3R1ZQUp6/BQnBt7Oo0JWBtc9qkv7JUDdcBJXPV5QWS5AyMPH
pqQ75Hitjsq/Fzu8eHtkKpFizcxGa9BZdkQjh4rSrtO1Kjs0Rd5DQtWSgkeVCCN09kN0ZsZ0ENY+Ip8QxSmyzt
sStkYXdULqpwz4JEXW9vz64eTbde4vQJ6pjHGarJf1gQNEc2XzhmI/prXLysWNqC7lZg7PUZUTrdegABTUzYCR
J1kWBRPm4qo0LN405c94QQd45a5kTgowHzEgLnAQI28x0M3A59TKC+ieNc6VF1PsTLpUw7PNI2VstX5jAuasCA
wEAAaMSMBAwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBCwUAA4IBAQCK6LGA01TR+rmU8p6yhAi4OkDN2b1
dbIL8l8iCMYopLCxx8xqq3ubZCOxqh1X2j6pgWzarb0b/MUix00IoUvNbFOxAW7PBZIKDLnm6LsckRxs1U32sC
9d1LOHe3WKBNB6GZALT1ewjh7hSbWjftlmcovq+6eVGA5cvf2u/2+TkKkyHV/NR394nXrdsdpvygwypEtXjetz
D7UT93Nuw3xcV8VIftIvHf9LjU7h+UjGmKXG9c15eYr3SzUmv6kyOI0Bvw14PWtsWGl0QdOSRvIBBrP4adCnGT
gjgjk9LTcO8B8FKrr+8lHGuc0bp4lIUToiUkGILXsiEeEg9WAqm+XqO"
  } ]
} ]
```

Get key

This method allows retrieving of key informations for a given alias.

Request

```
GET /services/rest/server-signing/key/certificate HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Host: localhost:8080
```

Response

```
HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 06:08:45 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
Transfer-Encoding: chunked
Content-Length: 2185

{
  "alias" : "certificate",
  "encryptionAlgo" : "RSA",
  "certificate" : {
    "encodedCertificate" :
"MIIC6jCCAdKgAwIBAgIGLtYU17tXMA0GCSqGSIb3DQEBCwUAMDAxGzAZBgNVBAMMElJvb3RTZWxmU2lnbmVkR
mFrZTERMA8GA1UECgwIRFNTLXRlc3QwHhcNMTcwNjA4MTEyNjAxWhcNNDcwNzA0MDc1NzI0WjAoMRMwEQYDVQQ
DDApTaWduZXJGYWtlMREwDwYDVQQKDAhEU1MtdGVzdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
MI3kZhtnipn+iiZHZ9ax8FlfE5Ow/cFwBTfAEb3R1ZQUp6/BQnBt7Oo0JWBtc9qkv7JUDdcBJXPV5QWS5AyMPH
pqQ75Hitjsq/Fzu8eHtkKpFizcxGa9BZdkQjh4rSrtO1Kjs0Rd5DQtWSgkeVCCN09kN0ZsZ0ENY+Ip8QxSmyzt
sStkYXdULqpwz4JEXW9vz64eTbde4vQJ6pjHGarJf1gQNEc2XzhmI/prXLysWNqC7lZg7PUZUTrdegABTUzYCR
J1kWBRPm4qo0LN405c94QQd45a5kTgowHzEgLnAQI28x0M3A59TKC+ieNc6VF1PsTLpUw7PNI2VstX5jAuasCA
wEAAaMSMBAwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBCwUAA4IBAQCK6LGA01TR+rmU8p6yhAi4OkDN2b1
dbIL8l8iCMYopLCxx8xqq3ubZCOxqh1X2j6pgWzarb0b/MUix00IoUvNbFOxAW7PBZIKDLnm6LsckRxs1U32sC
9d1LOHe3WKBNB6GZALT1ewjh7hSbWjftlmcovq+6eVGA5cvf2u/2+TkKkyHV/NR394nXrdsdpvygwypEtXjetz
D7UT93Nuw3xcV8VIftIvHf9LjU7h+UjGmKXG9c15eYr3SzUmv6kyOI0Bvw14PWtsWGl0QdOSRvIBBrP4adCnGT
gjgjk9LTcO8B8FKrr+8lHGuc0bp4lIUToiUkGILXsiEeEg9WAqm+XqO"
  },
  "certificateChain" : [ {
    "encodedCertificate" :
"MIIC6jCCAdKgAwIBAgIGLtYU17tXMA0GCSqGSIb3DQEBCwUAMDAxGzAZBgNVBAMMElJvb3RTZWxmU2lnbmVkR
mFrZTERMA8GA1UECgwIRFNTLXRlc3QwHhcNMTcwNjA4MTEyNjAxWhcNNDcwNzA0MDc1NzI0WjAoMRMwEQYDVQQ
DDApTaWduZXJGYWtlMREwDwYDVQQKDAhEU1MtdGVzdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
MI3kZhtnipn+iiZHZ9ax8FlfE5Ow/cFwBTfAEb3R1ZQUp6/BQnBt7Oo0JWBtc9qkv7JUDdcBJXPV5QWS5AyMPH
pqQ75Hitjsq/Fzu8eHtkKpFizcxGa9BZdkQjh4rSrtO1Kjs0Rd5DQtWSgkeVCCN09kN0ZsZ0ENY+Ip8QxSmyzt
sStkYXdULqpwz4JEXW9vz64eTbde4vQJ6pjHGarJf1gQNEc2XzhmI/prXLysWNqC7lZg7PUZUTrdegABTUzYCR
J1kWBRPm4qo0LN405c94QQd45a5kTgowHzEgLnAQI28x0M3A59TKC+ieNc6VF1PsTLpUw7PNI2VstX5jAuasCA
wEAAaMSMBAwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBCwUAA4IBAQCK6LGA01TR+rmU8p6yhAi4OkDN2b1
dbIL8l8iCMYopLCxx8xqq3ubZCOxqh1X2j6pgWzarb0b/MUix00IoUvNbFOxAW7PBZIKDLnm6LsckRxs1U32sC
9d1LOHe3WKBNB6GZALT1ewjh7hSbWjftlmcovq+6eVGA5cvf2u/2+TkKkyHV/NR394nXrdsdpvygwypEtXjetz
D7UT93Nuw3xcV8VIftIvHf9LjU7h+UjGmKXG9c15eYr3SzUmv6kyOI0Bvw14PWtsWGl0QdOSRvIBBrP4adCnGT
gjgjk9LTcO8B8FKrr+8lHGuc0bp4lIUToiUkGILXsiEeEg9WAqm+XqO"
  } ]
}
```

Sign

This method allows signing of given digests with a server side certificate.

Request

```
POST /services/rest/server-signing/sign/certificate/SHA256 HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 24

{
  "bytes" : "AQID"
}
```

Response

```
HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 06:08:46 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
Transfer-Encoding: chunked
Content-Length: 395

{
  "algorithm" : "RSA_SHA256",
  "value" :
"AZgLVQQLPQkPgRlfTNfTg3QlcDa0JTb0lS6kSteHxHLvjTmtKnRfYTVPZ0bupdPMVQIfuBt4OQv2zVtTbor+k
j1u7BaaeO50mXB8OMvo93F/ZmHPIff8VduPASOql7xc4TN73I6KoAn6ouYT0juxluQa9r79yvGo/qhoUwu9R/j
GfOfGPKNHbGVDqnG1rHX0qEWPKIYxetiTLnaIZGxuZ9p2vDzZRoEaTs0UWcFu8Yln9Xk8fe6hSxAQOncBXwQX8
LKAmZH4/QLsGuJwr+2FhsnC4slXi1TdXPzAlqLU38gmamK+QjqMTIPmQioLq2WLVhLye59dHvgvDChkTW3IZA=
="
}
```

REST validation service

DSS provides also a module for documents validation.

Validate a document

This service allows a signature validation (all formats/types) against a validation policy.

Request

```
POST /services/rest/validation/validateSignature HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 7365

{
  "signedDocument" : {
    "bytes" :
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48ZHM6U2lnbmF0dXJlIHhtbG5zOmRzPSJod
HRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjIiBJZD0iaWQtYWZkZTc4MjQzNjQ2OGRkNzRlZWIxODF
mN2NlMTEwZTEiPjxkczpTaWduZWRJbmZvPjxkczpDYW5vbmljYWxpemF0aW9uTWV0aG9kIEFsZ29yaXRobT0ia
HR0cDovL3d3dy53My5vcmcvMjAwMS8xMC94bWwtZXhjLWMxNG4jIi8+PGRzOlNpZ25hdHVyZU1ldGhvZCBBbGd
vcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZHNpZy1tb3JlI3JzYS1zaGEyNTYiLz48ZHM6U
mVmZXJlbmNlIElkPSJyLWlkLTEiIFR5cGU9IiIgVVJJPSJzYW1wbGUueG1sIj48ZHM6RGlnZXN0TWV0aG9kIEF
sZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhMjU2Ii8+PGRzOkRpZ2VzdFZhb
HVlPmtjREhPWmp3WmhWZnVEaHVoQ2VDRVJSbVlwVEg0Smo0Um1mVlZpMzFROWc9PC9kczpEaWdlc3RWYWx1ZT4
8L2RzOlJlZmVyZW5jZT48ZHM6UmVmZXJlbmNlIFR5cGU9Imh0dHA6Ly91cmkuZXRzaS5vcmcvMDE5MDMjU2lnb
mVkUHJvcGVydGllcyIgVVJJPSIjeGFkZXMtaWQtYWZkZTc4MjQzNjQ2OGRkNzRlZWIxODFmN2NlMTEwZTEiPjx
kczpUcmFuc2Zvcm1zPjxkczpUcmFuc2Zvcm0gQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzEwL
3htbC1leGMtYzE0biMiLz48L2RzOlRyYW5zZm9ybXM+PGRzOkRpZ2VzdE1ldGhvZCBBbGdvcml0aG09Imh0dHA
6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZW5jI3NoYTI1NiIvPjxkczpEaWdlc3RWYWx1ZT5EenR3TlRtUm9PQ
W02L2xNSThSeW01eFpQekl2TFlEem4vZWJZWWtQc3I0PTwvZHM6RGlnZXN0VmFsdWU+PC9kczpSZWZlcmVuY2U
+PC9kczpTaWduZWRJbmZvPjxkczpTaWduYXR1cmVWYWx1ZSBJZD0idmFsdWUtaWQtYWZkZTc4MjQzNjQ2OGRkN
zRlZWIxODFmN2NlMTEwZTEiPllBN3NFTnQzTjh1ZkxGTW5LcjM2cjBQcXpNaVkzUTBzKytJR1RFVUMwc3BheFV
2MGRIWk0wZC95bjNrcExxKTG9Va0k0TTNmbGo1V0duODNrZjA1QnFNMWtoc1g2MUdKemFGVFBHcG03YWtSUUtod
m9IMjV5eXFUWVhFU2xCY20wNGl6aUtoTE16WmpVZng0L0IxWkl5c3Y1cElCZ0oycjJvaTZqTG9wOXd3M2dlNGM
0WUpvYUsrU1hrNmh5VE5PY044UGpHZTYzV1lPVE5WUFFGdmphOEJud2crYTBiQnV3RCs4TjZmd2lnQ2RXNWEvN
ERKVWUvSjhNYjcwWkk4UG9Pem5HRGZpK1RQYmlJZVZtQ2JsNW1Vb1VnMlEveFlsdUpmTGgzdUdRQVhLQnZGNDV
vRElUlZlZm5OL0QvV3l0QUNsVVZEb1FTeXdlbW5qUHBxRjhlZz09PC9kczpTaWduYXR1cmVWYWx1ZT48ZHM6S
2V5SW5mbz48ZHM6WDUwOURhdGE+PGRzOlg1MDlDZXJ0aWZpY2F0ZT5NSUlEMURDQ0FyeWdBd0lCQWdJQkNqQU5
CZ2txaGtpRzl3MEJBUXNGQURCTk1SQXdEZ1lEVlFRRERBZG5iMjlrTFdOaE1Sa3dGd1lEVlFRS0RCQk9iM2RwY
m1FZ1UyOXNkWFJwYjI1ek1SRXdEd1lEVlFRTERBaFFTMGt0VkVWVFZERUxNQWtHQTFVRUJoTUNURlV3SGhjTk1
UWXhNREkyTURjMU5ETXhXaGNOTVRnd09ESTJNRGMxTkRNeFdqQlBNUkl3RUFZRFZRUUREQWxuYjI5a0xYVnpaW
El4R1RBWEJnTlZCQW9NRUU1dmQybHVZU0JUYjJ4MWRHbHZibk14RVRBUEJnTlZCQXNNQ0ZCTFNTMVVSVk5VTVF
zd0NRWURWUVFHRXdKTVZUQ0NBU0l3RFFZSktvWklodmNOQVFFQkJRQURnZ0VQQURDQ0FRb0NnZ0VCQUxSQ1VJU
VpidzNuU2RMcCtCOWN6RUNncFpra1E1eFY0ZzlNLzd3bGc5N29DQ2Y3VUVoOUJBMWQrellqc3p2K0JKMWJKWlB
nYW4yMTQ0QXZnc29HSmZiNlVJeVZXNGdrbFVnSWwxYXJVdm9uK1drS25zZUZ1UU9mSnlqU0ZVREl3bnV2cDBoe
mNKWEhYUm1MZG15aCtuKzZOTUgwb201dFZvU2ZRcnRCVmlDTGVTTVZ6dUQ1RVBqMG1JUmN4OTFwTDM4ZTNGTlR
XN05hR1pMZWV6dUZ1Ui9xN3o5M2xMa3ZaNFZBTU5HR0x2SVhPWWVSQlpNeVBocEJaNEwzQThJM0VFbEtXSC8xT
HdpaVhUVFNHMXNNNld2TVRWYmYydmJkNDduWlJRQTJtU3BOR2pRb3VPQUVyZmVWVlVxeklDZ2hRQ0hSR09OdVN
MRy9IZnFGSGI0aldnMENBd0VBQWFPQnZEQ0J1VEFPQmdOVkhROEJBZjhFQkFNQ0JrQXdnWWNHQ0NzR0FRVUZCd
0VCQkhzd2VUQTVCZ2dyQmdFRkJRY3dBWVl0YUhSMGNEb3ZMMlJ6Y3k1dWIzZHBibUV1YkhVdmNHdHBMV1poWTN
SdmNua3ZiMk56Y0M5bmIyOWtMV05oTUR3R0NDc0dBUVVGQnpBQ2hqQm9kSFJ3T2k4dlpITnpMbTV2ZDJsdVlTN
XNkUzl3YTJrdFptRmpkRzl5ZVM5amNuUXZaMjl2WkMxallTNWpjblF3SFFZRFZSME9CQllFRk4ycEhELzdQZWZ
tQlQ4b1gyOVpoV3kvT0hKMU1BMEdDU3FHU0liM0RRRUJDd1VBQTRJQkFRQkszVk9MaERJVldLb0ZycmhoV3phZ
GR0azZYUXRjd1JvTlBWU3NpL2dPcnpzZE03MEEzMXhJVHc3WWZMaHBvVkExeG83b3ZIbGRwTGxocXk5bzV3aDI
4MnlDcHFCVUF0Z3JTa0RHb2crSzdDTDZnVXBybFlpWnVHWnJ0ZzJYM2ZIUzJVc3g0WkozdElqNndWZWNERVVxS
VNGZkZUMkVzbTBRWFVuZ0lLRk1sOTVYZ210dzJ3eFhiT3pVZURkNERJUHJ2K21XNXBvQVdyNkl0c1YrSDJWUSt
```

```
aTC9rQm53V0hqU1RPYUdGaXNxWFkvYUgvMVB0QlhBKzE1K1lJV2VtSkJTdjNrRGFGek9YQUV0UjlaSThsWU9KY
XJuWTdBeS9hTjZiOXVHZmZyYm8vaFZBY0w0V0RkaGtiQk4zbTh3K2c3NkxvQVhOZUVldTA0QS8weExaelVCPC9
kczpYNTA5Q2VydGlmaWNhdGU+PGRzOlg1MDlDZXJ0aWZpY2F0ZT5NSUlENmpDQ0F0S2dBd0lCQWdJQkJEQU5CZ
2txaGtpRzl3MEJBUXNGQURCTk1SQXdEZ1lEVlFRRERBZHliMjkwTFdOaE1Sa3dGd1lEVlFRS0RCQk9iM2RwYm1
FZ1UyOXNkWFJwYjI1ek1SRXdEd1lEVlFRTERBaFFTMGt0VkVWVFZERUxNQWtHQTFVRUJoTUNURlV3SGhjTk1UW
XhNREkyTURjMU5ETXdXaGNOTVRnd09ESTJNRGMxTkRNd1dqQk5NUkF3RGdZRFZRUUREQWRuYjI5a0xXTmhNUmt
3RndZRFZRUUtEQkJPYjNkcGJtRWdVMjlzZFhScGIyNXpNUkV3RHdZRFZRUUxEQWhRUzBrdFZFVlRWREVMTUFrR
0ExVUVCaE1DVEZVd2dnRWlNQTBHQ1NxR1NJYjNEUUVCQVFVQUE0SUJEd0F3Z2dFS0FvSUJBUUNiYmw1c0tCQ2p
TQjhUTWRhY3lteC9XZk9qTVcxZ2lJalZKUlkyOEpiTldrQ1ZtdHpnbWl0Z2hmclBRUGx1ZXUwRFRhbGJEa3JTU
3loQ3Z6elBTR1B3Q0ZPYWhGL243aFFhMUYzVWFIU3hUS3JGQzVuT3dkTHp4S1JPM1dqVnRJR1JTWDJrdjFGZlV
wUXk1RXl5K3JzZlN6SjU5ZFU1WlpkV3BkYUR1RHhWVnlEZXIzRUl5Q2JHNy81SDlNRDRZdXpOcGVURldtTTZjV
VNUMDc5NlhEbGJFeFNUVEdRWEZKQTIrQ0NzeTlEWG5KYThuejBGRThmbWN2UUhlVTZrOVFicHpHak1kM0RXbEU
2bm83VWRDWUQxSDA0K3VzQnA1aGhDckFCNjcwTmRvVHJOVG1HTkFGdDRKVDB2aXRqS0hxOUtFSWQ2TGhkY20yV
Gc5M2REY1dGdEFnTUJBQUdqZ2RRd2dkRXdEZ1lEVlIwUEFRSC9CQVFEQWdlQU1FRUdBMVVkSHdRNk1EZ3dOcUE
wb0RLR01HaDBkSEE2THk5a2MzTXVibTkzYVc1aExteDFMM0JyYVMxbVlXTjBiM0o1TDJOeWJDOXliMjkwTFdOa
ExtTnliREJNQmdnckJnRUZCUWNCQVFSQU1ENHdQQVlJS3dZQkJRVUhNQUtHTUdoMGRIQTZMeTlrYzNNdWJtOTN
hVzVoTG14MUwzQnJhUzFtWVdOMGIzSjVMMk55ZEM5eWIyOTBMV05oTG1OeWREQWRCZ05WSFE0RUZnUVUrMnRGc
XBOZTNHMjNZUjh5cUJaSWlWV1MzdlV3RHdZRFZSMFRBUUgvQkFVd0F3RUIvekFOQmdrcWhraUc5dzBCQVFzRkF
BT0NBUUVBRStOdWQwNVhHT002RkVaSFdUYzgrYm16LzZCMFhRWE41NjRLV0JCaGNoOWk1R2FkanFwU3NldmtuK
3RlTHE1bTZDTG8zZTRsWDJkSjdoc1BBdnlhTHFPSXB6ZzQ5VEdkaWIxbk9CMk83NCt5QWhUOHY5Rlp0SDFFQ0h
YeFlzdXlTR0lLdmQrTDVJakpUaXMzbGw0dlU4Rkh6eVJsTTlJUW53WlI1MDZqRmNKZUdsT2d5WmgrVUxXblJOR
UV3cU44RFRGMkQwWG9nWUJzckN4Q0JqMFBwYUpGcnV2RVFxcFV1dVlnMTRSMURKRmFoTHdxVllTT0Q1Z1BobUE
wSFI0ejNHRjNqSFN6MGk5alhTVE9zVWNka3ZVSnkwdElPbnVqc1VFa2czSDZXZzNsejhUdzNJYzdWMU5IYitNQ
zVLNFp2WCs1U1l5dTArZXI3YkZzY0lyWVp3PT08L2RzOlg1MDlDZXJ0aWZpY2F0ZT48L2RzOlg1MDlEYXRhPjw
vZHM6S2V5SW5mbz48ZHM6T2JqZWN0Pjx4YWRlczpRdWFsaWZ5aW5nUHJvcGVydGllcyB4bWxuczp4YWRlcz0ia
HR0cDovL3VyaS5ldHNpLm9yZy8wMTkwMy92MS4zLjIjIiBUYXJnZXQ9IiNpZC1hZmRlNzgyNDM2NDY4ZGQ3NGV
lYjE4MWY3Y2UxMTBlMSI+PHhhZGVzOlNpZ25lZFByb3BlcnRpZXMgSWQ9InhhZGVzLWlkLWFmZGU3ODI0MzY0N
jhkZDc0ZWViMTgxZjdjZTExMGUxIj48eGFkZXM6U2lnbmVkU2lnbmF0dXJlUHJvcGVydGllcz48eGFkZXM6U2l
nbmluZ1RpbWU+MjAxNy0wOS0yOFQxMTowOTowNFo8L3hhZGVzOlNpZ25pbmdUaW1lPjx4YWRlczpTaWduaW5nQ
2VydGlmaWNhdGVWMj48eGFkZXM6Q2VydD48eGFkZXM6Q2VydERpZ2VzdD48ZHM6RGlnZXN0TWV0aG9kIEFsZ29
yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI3NoYTEiLz48ZHM6RGlnZXN0VmFsdWU+Y
ytWb2hnMGpJY1o0VVFTV2VnbENnMG9HTldzPTwvZHM6RGlnZXN0VmFsdWU+PC94YWRlczpDZXJ0RGlnZXN0Pjx
4YWRlczpJc3N1ZXJTZXJpYWxWMj5NRll3VWFSUE1FMHhFREFPQmdOVkJBTU1CMmR2YjJRdFkyRXhHVEFYQmdOV
kJBb01FRTV2ZDJsdVlTQlRiMngxZEdsdmJuTXhFVEFQQmdOVkJBc01DRkJMU1MxVVJWTlVNUXN3Q1FZRFZRUUd
Fd0pNVlFJQkNnPT08L3hhZGVzOklzc3VlclNlcmlhbFYyPjwveGFkZXM6Q2VydD48L3hhZGVzOlNpZ25pbmdDZ
XJ0aWZpY2F0ZVYyPjwveGFkZXM6U2lnbmF0dXJlUHJvcGVydGllcz48eGFkZXM6U2lnbmVkRGF0YU9
iamVjdFByb3BlcnRpZXM+PHhhZGVzOkRhdGFPYmplY3RGb3JtYXQgT2JqZWN0UmVmZXJlbmNlPSIjci1pZC0xI
j48eGFkZXM6TWltZVR5cGU+dGV4dC94bWw8L3hhZGVzOk1pbWVUeXBlPjwveGFkZXM6RGF0YU9iamVjdEZvcm1
hdD48L3hhZGVzOlNpZ25lZERhdGFPYmplY3RQcm9wZXJ0aWVzPjwveGFkZXM6U2lnbmVkUHJvcGVydGllcz48L
3hhZGVzOlF1YWxpZnlpbmdQcm9wZXJ0aWVzPjwvZHM6T2JqZWN0PjwvZHM6U2lnbmF0dXJlPg==",
    "digestAlgorithm" : null,
    "name" : "xades-detached.xml"
  },
  "originalDocuments" : [ {
    "bytes" :
"77u/PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4NCjxoOnRhYmxlIHhtbG5zOmg9Imh0d
HA6Ly93d3cudzMub3JnL1RSL2h0bWw0LyI+DQoJPGg6dHI+DQoJCTxoOnRkPkhlbGxvPC9oOnRkPg0KCQk8aDp
0ZD5Xb3JsZDwvaDp0ZD4NCgk8L2g6dHI+DQo8L2g6dGFibGU+",
    "digestAlgorithm" : null,
    "name" : "sample.xml"
  } ],
```

```
  "policy" : null,
  "signatureId" : null
}
```

Response


```
HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 06:08:45 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
Transfer-Encoding: chunked
Content-Length: 48309

{
  "validationReportaDataHandler" :
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiIHN0YW5kYWxvbmU9InllcyI/Pgo8VmFsaWRhd
GlvblJlcG9ydCB4bWxucz0iaHR0cDovL3VyaS5ldHNpLm9yZy8xOTEwMi92MS4yLjEjIiB4bWxuczpuczI9Imh
0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyMiIHhtbG5zOm5zMz0iaHR0cDovL3VyaS5ldHNpLm9yZ
y8wMTkwMy92MS4zLjIjIiB4bWxuczpuczQ9Imh0dHA6Ly91cmkuZXRzaS5vcmcvMDIyMzEvdjIjIj4KICAgIDx
TaWduYXR1cmVWYWxpZGF0aW9uUmVwb3J0PgogICAgICAgIDxTaWduYXR1cmVJZGVudGlmaWVyIGlkPSJTLURFM
UE3QjMyNDhGNzBCQjYzRTg5RDFFMDIxODMzMDM3M0UyOTAwNUE4ODY3MEVFOTc1OEZCNTRFQ0VBQkU1RDciPgo
gICAgICAgICAgICA8RGlnZXN0QWxnQW5kVmFsdWU+CiAgICAgICAgICAgICAgICA8bnMyOkRpZ2VzdE1ldGhvZ
CBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZW5jI3NoYTI1NiIvPgogICAgICAgICA
gICAgICAgPG5zMjpEaWdlc3RWYWx1ZT5EenR3TlRtUm9PQW02L2xNSThSeW01eFpQekl2TFlEem4vZWJZWWtQc
3I0PTwvbnMyOkRpZ2VzdFZhbHVlPgogICAgICAgICAgICA8L0RpZ2VzdEFsZ0FuZFZhbHVlPgogICAgICAgICA
gICA8bnMyOlNpZ25hdHVyZVZhbHVlPllBN3NFTnQzTjh1ZkxGTW5LcjM2cjBQcXpNaVkzUTBzKytJR1RFVUMwc
3BheFV2MGRIWk0wZC95bjNrcExKTG9Va0k0TTNmbGo1V0duODNrZjA1QnFNMWtoc1g2MUdKemFGVFBHcG03YWt
SUUtodm9IMjV5eXFUWVhFU2xCY20wNGl6aUtoTE16WmpVZng0L0IxWkl5c3Y1cElCZ0oycjJvaTZqTG9wOXd3M
2dlNGM0WUpvYUsrU1hrNmh5VE5PY044UGpHZTYzV1lPVE5WUFFGdmphOEJud2crYTBiQnV3RCs4TjZmd2lnQ2R
XNWEvNERKVWUvSjhNYjcwWkk4UG9Pem5HRGZpK1RQYmlJZVZtQ2JsNW1Vb1VnMlEveFlsdUpmTGgzdUdRQVhLQ
nZGNDVvRElIUlZlZm5OL0QvV3l0QUNsVVZEb1FTeXdlbW5qUHBxRjhlZz09PC9uczI6U2lnbmF0dXJlVmFsdWU
+CiAgICAgICAgICAgIDxIYXNoT25seT5mYWxzZTwvSGFzaE9ubHk+CiAgICAgICAgICAgIDxEb2NIYXNoT25se
T5mYWxzZTwvRG9jSGFzaE9ubHk+CiAgICAgICAgICAgIDxEQUlkZW50aWZpZXI+aWQtYWZkZTc4MjQzNjQ2OGR
kNzRlZWIxODFmN2NlMTEwZTE8L0RBSWRlbnRpZmllcj4KICAgICAgICA8L1NpZ25hdHVyZUlkZW50aWZpZXI+C
iAgICAgICAgPFZhbGlkYXRpb25Db25zdHJhaW50c0V2YWx1YXRpb25SZXBvcnQ+CiAgICAgICAgICAgIDxWYWx
pZGF0aW9uQ29uc3RyYWludD4KICAgICAgICAgICAgICAgIDxWYWxpZGF0aW9uQ29uc3RyYWludElkZW50aWZpZ
XI+dXJuOmNlZjpkc3M6YmJiOmZvcm1hdENoZWNraW5nPC9WYWxpZGF0aW9uQ29uc3RyYWludElkZW50aWZpZXI
+CiAgICAgICAgICAgICAgICA8Q29uc3RyYWludFN0YXR1cz4KICAgICAgICAgICAgICAgICAgICA8U3RhdHVzP
nVybjpldHNpOjAxOTEwMjpjb25zdHJhaW50U3RhdHVzOmFwcGxpZWQ8L1N0YXR1cz4KICAgICAgICAgICAgICA
gIDwvQ29uc3RyYWludFN0YXR1cz4KICAgICAgICAgICAgICAgIDxWYWxpZGF0aW9uU3RhdHVzPgogICAgICAgI
CAgICAgICAgICAgIDxNYWluSW5kaWNhdGlvbj51cm46ZXRzaTowMTkxMDI6bWFpbmluZGljYXRpb246cGFzc2V
kPC9NYWluSW5kaWNhdGlvbj4KICAgICAgICAgICAgICAgIDwvVmFsaWRhdGlvblN0YXR1cz4KICAgICAgICAgI
CAgPC9WYWxpZGF0aW9uQ29uc3RyYWludD4KICAgICAgICAgICAgPFZhbGlkYXRpb25Db25zdHJhaW50PgogICA
gICAgICAgICAgPFZhbGlkYXRpb25Db25zdHJhaW50SWRlbnRpZmllcj51cm46Y2VmOmRzczpiYmI6aWRlb
nRpZmljYXRpb25PZlRoZVNpZ25pbmdDZXJ0aWZpY2F0ZTwvVmFsaWRhdGlvbkNvbnN0cmFpbnRJZGVudGlmaWV
```

```
yPgogICAgICAgICAgICAgICAgPENvbnN0cmFpbnRTdGF0dXM+CiAgICAgICAgICAgICAgICAgICAgPFN0YXR1c
z51cm46ZXRzaTowMTkxMDI6Y29uc3RyYWludFN0YXR1czphcHBsaWVkPC9TdGF0dXM+CiAgICAgICAgICAgICA
gICA8L0NvbnN0cmFpbnRTdGF0dXM+CiAgICAgICAgICAgICAgICA8VmFsaWRhdGlvblN0YXR1cz4KICAgICAgI
CAgICAgICAgICAgICA8TWFpbkluZGljYXRpb24+dXJuOmV0c2k6MDE5MTAyOm1haW5pbmRpY2F0aW9uOnBhc3N
lZDwvTWFpbkluZGljYXRpb24+CiAgICAgICAgICAgICAgICA8L1ZhbGlkYXRpb25TdGF0dXM+CiAgICAgICAgI
CAgIDwvVmFsaWRhdGlvbkNvbnN0cmFpbnQ+CiAgICAgICAgICAgIDxWYWxpZGF0aW9uQ29uc3RyYWludD4KICA
gICAgICAgICAgICAgIDxWYWxpZGF0aW9uQ29uc3RyYWludElkZW50aWZpZXI+dXJuOmNlZjpkc3M6YmJiOnZhb
GlkYXRpb25Db250ZXh0SW5pdGlhbGl6YXRpb248L1ZhbGlkYXRpb25Db25zdHJhaW50SWRlbnRpZmllcj4KICA
gICAgICAgICAgICAgIDxDb25zdHJhaW50U3RhdHVzPgogICAgICAgICAgICAgICAgICAgIDxTdGF0dXM+dXJuO
mV0c2k6MDE5MTAyOmNvbnN0cmFpbnRTdGF0dXM6YXBwbGllZDwvU3RhdHVzPgogICAgICAgICAgICAgICAgPC9
Db25zdHJhaW50U3RhdHVzPgogICAgICAgICAgICAgICAgPFZhbGlkYXRpb25TdGF0dXM+CiAgICAgICAgICAgI
CAgICAgICAgPE1haW5JbmRpY2F0aW9uPnVybjpldHNpOjAxOTEwMjptYWluaW5kaWNhdGlvbjpwYXNzZWQ8L01
haW5JbmRpY2F0aW9uPgogICAgICAgICAgICAgICAgPC9WYWxpZGF0aW9uU3RhdHVzPgogICAgICAgICAgICA8L
1ZhbGlkYXRpb25Db25zdHJhaW50PgogICAgICAgICAgICA8VmFsaWRhdGlvbkNvbnN0cmFpbnQ+CiAgICAgICA
gICAgICA8VmFsaWRhdGlvbkNvbnN0cmFpbnRJZGVudGlmaWVyPnVybjpjZWY6ZHNzOmJiYjpjcnlwdG9nc
mFwaGljVmVyaWZpY2F0aW9uPC9WYWxpZGF0aW9uQ29uc3RyYWludElkZW50aWZpZXI+CiAgICAgICAgICAgICA
gICA8Q29uc3RyYWludFN0YXR1cz4KICAgICAgICAgICAgICAgICAgICA8U3RhdHVzPnVybjpldHNpOjAxOTEwM
jpjb25zdHJhaW50U3RhdHVzOmFwcGxpZWQ8L1N0YXR1cz4KICAgICAgICAgICAgICAgIDwvQ29uc3RyYWludFN
0YXR1cz4KICAgICAgICAgICAgICAgIDxWYWxpZGF0aW9uU3RhdHVzPgogICAgICAgICAgICAgICAgICAgIDxNY
WluSW5kaWNhdGlvbj51cm46ZXRzaTowMTkxMDI6bWFpbmluZGljYXRpb246cGFzc2VkPC9NYWluSW5kaWNhdGl
vbj4KICAgICAgICAgICAgICAgIDwvVmFsaWRhdGlvblN0YXR1cz4KICAgICAgICAgICAgPC9WYWxpZGF0aW9uQ
29uc3RyYWludD4KICAgICAgICAgICAgPFZhbGlkYXRpb25Db25zdHJhaW50PgogICAgICAgICAgICAgICAgPFZ
hbGlkYXRpb25Db25zdHJhaW50SWRlbnRpZmllcj51cm46Y2VmOmRzczpiYmI6c2lnbmF0dXJlQWNjZXB0YW5jZ
VZhbGlkYXRpb248L1ZhbGlkYXRpb25Db25zdHJhaW50SWRlbnRpZmllcj4KICAgICAgICAgICAgICAgIDxDb25
zdHJhaW50U3RhdHVzPgogICAgICAgICAgICAgICAgICAgIDxTdGF0dXM+dXJuOmV0c2k6MDE5MTAyOmNvbnN0c
mFpbnRTdGF0dXM6YXBwbGllZDwvU3RhdHVzPgogICAgICAgICAgICAgICAgPC9Db25zdHJhaW50U3RhdHVzPgo
gICAgICAgICAgICAgICAgPFZhbGlkYXRpb25TdGF0dXM+CiAgICAgICAgICAgICAgICAgICAgPE1haW5JbmRpY
2F0aW9uPnVybjpldHNpOjAxOTEwMjptYWluaW5kaWNhdGlvbjpwYXNzZWQ8L01haW5JbmRpY2F0aW9uPgogICA
gICAgICAgICAgICAgPC9WYWxpZGF0aW9uU3RhdHVzPgogICAgICAgICAgICA8L1ZhbGlkYXRpb25Db25zdHJha
W50PgogICAgICAgICAgICA8VmFsaWRhdGlvbkNvbnN0cmFpbnQ+CiAgICAgICAgICAgICAgICA8VmFsaWRhdGl
vbkNvbnN0cmFpbnRJZGVudGlmaWVyPnVybjpjZWY6ZHNzOmJiYjp4NTA5Q2VydGlmaWNhdGVWYWxpZGF0aW9uP
C9WYWxpZGF0aW9uQ29uc3RyYWludElkZW50aWZpZXI+CiAgICAgICAgICAgICAgICA8Q29uc3RyYWludFN0YXR
1cz4KICAgICAgICAgICAgICAgICAgICA8U3RhdHVzPnVybjpldHNpOjAxOTEwMjpjb25zdHJhaW50U3RhdHVzO
mFwcGxpZWQ8L1N0YXR1cz4KICAgICAgICAgICAgICAgIDwvQ29uc3RyYWludFN0YXR1cz4KICAgICAgICAgICA
gICAgIDxWYWxpZGF0aW9uU3RhdHVzPgogICAgICAgICAgICAgICAgICAgIDxNYWluSW5kaWNhdGlvbj51cm46Z
XRzaTowMTkxMDI6bWFpbmluZGljYXRpb246aW5kZXRlcm1pbmF0ZTwvTWFpbkluZGljYXRpb24+CiAgICAgICA
gICAgICAgICAgICAgPFN1YkluZGljYXRpb24+dXJuOmV0c2k6MDE5MTAyOnN1YmluZGljYXRpb246Tk9fQ0VSV
ElGSUNBVEVfQ0hBSU5fRk9VTkQ8L1N1YkluZGljYXRpb24+CiAgICAgICAgICAgICAgICA8L1ZhbGlkYXRpb25
TdGF0dXM+CiAgICAgICAgICAgIDwvVmFsaWRhdGlvbkNvbnN0cmFpbnQ+CiAgICAgICAgICAgIDxWYWxpZGF0a
W9uQ29uc3RyYWludD4KICAgICAgICAgICAgICAgIDxWYWxpZGF0aW9uQ29uc3RyYWludElkZW50aWZpZXI+dXJ
uOmNlZjpkc3M6YmJiOnBhc3RTaWduYXR1cmVWYWxpZGF0aW9uPC9WYWxpZGF0aW9uQ29uc3RyYWludElkZW50a
WZpZXI+CiAgICAgICAgICAgICAgICA8Q29uc3RyYWludFN0YXR1cz4KICAgICAgICAgICAgICAgICAgICA8U3R
hdHVzPnVybjpldHNpOjAxOTEwMjpjb25zdHJhaW50U3RhdHVzOmRpc2FibGVkPC9TdGF0dXM+CiAgICAgICAgI
CAgICAgICA8L0NvbnN0cmFpbnRTdGF0dXM+CiAgICAgICAgICAgIDwvVmFsaWRhdGlvbkNvbnN0cmFpbnQ+CiA
gICAgICAgICAgIDxWYWxpZGF0aW9uQ29uc3RyYWludD4KICAgICAgICAgICAgICAgIDxWYWxpZGF0aW9uQ29uc
3RyYWludElkZW50aWZpZXI+dXJuOmNlZjpkc3M6YmJiOnBhc3RDZXJ0aWZpY2F0ZVZhbGlkYXRpb248L1ZhbGl
kYXRpb25Db25zdHJhaW50SWRlbnRpZmllcj4KICAgICAgICAgICAgICAgIDxDb25zdHJhaW50U3RhdHVzPgogI
CAgICAgICAgICAgICAgICAgIDxTdGF0dXM+dXJuOmV0c2k6MDE5MTAyOmNvbnN0cmFpbnRTdGF0dXM6ZGlzYWJ
sZWQ8L1N0YXR1cz4KICAgICAgICAgICAgICAgIDwvQ29uc3RyYWludFN0YXR1cz4KICAgICAgICAgICAgPC9WY
WxpZGF0aW9uQ29uc3RyYWludD4KICAgICAgICAgICAgPFZhbGlkYXRpb25Db25zdHJhaW50PgogICAgICAgICA
gICAgICAgPFZhbGlkYXRpb25Db25zdHJhaW50SWRlbnRpZmllcj51cm46Y2VmOmRzczpiYmI6dmFsaWRhdGlvb
```

lRpbWVTbGlkaW5nPC9WYWxpZGF0aW9uQ29uc3RyYWludElkZW50aWZpZXI+CiAgICAgICAgICAgICAgICAgICA8Q29
uc3RyYWludFN0YXR1cz4KICAgICAgICAgICAgICAgICAgICAgICA8U3RhdHVzPnVybjpldHNpOjAxOTEwMjpjb25zd
HJhaW50U3RhdHVzOmRpc2FibGVkPC9TdGF0dXM+CiAgICAgICAgICAgICAgICAgICA8L0NvbnN0cmFpbnRTdGF0dXM
+CiAgICAgICAgICAgICAgIDwvVmFsaWRhdGlvbkNvbnN0cmFpbnQ+CiAgICAgICAgICAgPC9WYWxpZGF0aW9uQ29uc3RyY
WludHNFdmFsdWF0aW9uUmVwb3J0PgogICAgICAgICAgIDxWYWxpZGF0aW9uVGltZUluZm8+CiAgICAgICAgICAgIDx
WYWxpZGF0aW9uVGltZT4yMDE5LTA4LTIzVDA2OjA4OjQ1WjwvVmFsaWRhdGlvblRpbWU+CiAgICAgICAgICAgI
DxCZXN0U2lnbmF0dXJlVGltZT4KICAgICAgICAgICAgICAgIDxQT0VUaW1lPjIwMTktMDgtMjNUMDY6MDg6NDV
aPC9QT0VUaW1lPgogICAgICAgICAgICAgICAgPFR5cGVPZlByb29mPnVybjpldHNpOjAxOTEwMjpwb2V0eXBlO
nZhbGlkYXRpb248L1R5cGVPZlByb29mPgogICAgICAgICAgICA8L0Jlc3RTaWduYXR1cmVUaW1lPgogICAgICA
gIDwvVmFsaWRhdGlvblRpbWVJbmZvPgogICAgICAgIDxTaWduZXJzRG9jdW1lbnQ+CiAgICAgICAgICAgIDxEa
Wdlc3RBbGdBbmRWYWx1ZT4KICAgICAgICAgICAgICAgIDxuczI6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR
0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhMjU2Ii8+CiAgICAgICAgICAgICAgICA8bnMyOkRpZ
2VzdFZhbHVlPmtjREhPWmp3WmhWZnVEaHVoQ2VDRVJSbVlwVEg0Smo0Um1mVlZpMzFROWc9PC9uczI6RGlnZXN
0VmFsdWU+CiAgICAgICAgICAgIDwvRGlnZXN0QWxnQW5kVmFsdWU+CiAgICAgICAgICAgIDxTaWduZXJzRG9jd
W1lbnRSZWYgVk9SZWZlcmVuY2U9IkQtQzU4QzgwQTgwNTMwRTBGMzQ5QkMzMkRFRjUwMjgwRDc0QzRCN0VCRjI
1MjgwNDQwMTgxMTY3QTJGMUEwQjMxRCIvPgogICAgICAgIDwvU2lnbmVyc0RvY3VtZW50PgogICAgICAgIDxTa
WduYXR1cmVBdHRyaWJ1dGVzPgogICAgICAgICAgICA8U2lnbmluZ1RpbWUgU2lnbmVkPSJ0cnVlIj4KICAgICA
gICAgICAgIDxUaW1lPjIwMTctMDktMjhUMTE6MDk6MDRaPC9UaW1lPgogICAgICAgICAgICA8L1NpZ25pb
mdUaW1lPgogICAgICAgICAgICA8U2lnbmluZ0NlcnRpZmljYXRlIFNpZ25lZD0idHJ1ZSI+CiAgICAgICAgICA
gICAgICA8QXR0cmlidXRlT2JqZWN0IFZPUmVmZXJlbmNlPSJDLUYwRkYwQjQ1MTREMzE2MzA0RjI4MTdEQkEwQ
kZCMDVERURCOTg1MjdDMEU0N0M3M0U4RDhGREZFMTZERjI2N0UiLz4KICAgICAgICAgICAgICAgIDxDZXJ0SUQ
+CiAgICAgICAgICAgICAgICAgICAgPFg1MDlJc3N1ZXJTZXJpYWw+TUZZd1VhUlBNRTB4RURBT0JnTlZCQU1NQ
jJkmIyUXRZMkV4R1RBWEJnTlZCQW9NRUU1dmQybHVZU0JUYjJ4MWRHbHZibk14RVRBUEJnTlZCQXNNQ0ZCTFN
TMVVSVk5VTVFzd0NRWURWUVFHRXdKTVZRSUJDZz09PC9YNTA5SXNzdWVyU2VyaWFsPgogICAgICAgICAgICAgI
CAgICAgIDxuczI6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWx
kc2lnI3NoYTEiLz4KICAgICAgICAgICAgICAgICAgICA8bnMyOkRpZ2VzdFZhbHVlPmMrVm9oZzBqSWNNaFVRU
1dlZ2xDZzBvR05Xcz08L25zMjpEaWdlc3RWYWx1ZT4KICAgICAgICAgICAgICAgIDwvQ2VydElEPgogICAgICA
gICAgICA8L1NpZ25pbmdDZXJ0aWZpY2F0ZT4KICAgICAgICAgICAgPERhdGFPYmplY3RGb3JtYXQgU2lnbmVkP
SJ0cnVlIj4KICAgICAgICAgICAgICAgIDxNaW1lVHlwZT50ZXh0L3htbDwvTWltZVR5cGU+CiAgICAgICAgICA
gIDwvRGF0YU9iamVjdEZvcm1hdD4KICAgICAgICA8L1NpZ25hdHVyZUF0dHJpYnV0ZXM+CiAgICAgICAgPFNpZ
25lckluZm9ybWF0aW9uPgogICAgICAgICAgICA8U2lnbmVyQ2VydGlmaWNhdGUgVk9SZWZlcmVuY2U9IkMtRjB
GRjBCNDUxNEQzMTYzMDRGMjgxN0RCQTBCRkIwNURFREI5ODUyN0MwRTQ3QzczRThEOEZERkUxNkRGMjY3RSIvP
gogICAgICAgICAgICA8U2lnbmVyPmdvb2QtdXNlcjwvU2lnbmVyPgogICAgICAgIDwvU2lnbmVySW5mb3JtYXR
pb24+CiAgICAgICAgPFNpZ25hdHVyZVF1YWxpdHk+CiAgICAgICAgICAgIDxTaWduYXR1cmVRdWFsaXR5SW5mb
3JtYXRpb24+dXJuOmNlZjpkc3M6c2lnbmF0dXJlUXVhbGlmaWNhdGlvbjpub3RBcHBsaWNhYmxlPC9TaWduYXR
1cmVRdWFsaXR5SW5mb3JtYXRpb24+CiAgICAgICAgPC9TaWduYXR1cmVRdWFsaXR5PgogICAgICAgIDxTaWduY
XR1cmVWYWxpZGF0aW9uUHJvY2Vzcz4KICAgICAgICAgICAgPFNpZ25hdHVyZVZhbGlkYXRpb25Qcm9jZXNzSUQ
+dXJuOjpldHNpOjAxOTEwMjp2YWxpZGF0aW9ucHJvY2VzczpMVEE8L1NpZ25hdHVyZVZhbGlkYXRpb25Qcm9jZ
XNzSUQ+CiAgICAgICAgPC9TaWduYXR1cmVWYWxpZGF0aW9uUHJvY2Vzcz4KICAgICAgICA8U2lnbmF0dXJlVmF
saWRhdGlvblN0YXR1cz4KICAgICAgICAgICAgPE1haW5JbmRpY2F0aW9uPnVybjpldHNpOjAxOTEwMjptYWlua
W5kaWNhdGlvbjpubRldGVybWluYXRlPC9NYWluSW5kaWNhdGlvbj4KICAgICAgICAgICAgPFN1YkluZGljYXR
pb24+dXJuOmV0c2k6MDE5MTAyOnN1YmluZGljYXRpb246Tk9fQ0VSVElGSUNBVEVfQ0hBSU5fRk9VTkQ8L1N1Y
kluZGljYXRpb24+CiAgICAgICAgICAgIDxBc3NvY2lhdGVkVmFsaWRhdGlvblJlcG9ydERhdGE+CiAgICAgICA
gICAgICAgICA8Q2VydGlmaWNhdGVDaGFpbj4KICAgICAgICAgICAgICAgICAgICA8U2lnbmluZ0NlcnRpZmljY
XRlIFZPUmVmZXJlbmNlPSJDLUYwRkYwQjQ1MTREMzE2MzA0RjI4MTdEQkEwQkZCMDVERURCOTg1MjdDMEU0N0M
3M0U4RDhGREZFMTZERjI2N0UiLz4KICAgICAgICAgICAgICAgICAgICA8SW50ZXJtZWRpYXRlQ2VydGlmaWNhd
GUgVk9SZWZlcmVuY2U9IkMtNkYzNURFMzk2NUI5QTY5QkMzNjYxRDFBMzU1QjBBRTYwOTA3QURCNzQxQ0MxOTE
xRUZEMEYzQkU3MkQ2QTZFOSIvPgogICAgICAgICAgICAgICAgICAgPC9DZXJ0aWZpY2F0ZUNoYWluPgogICAgICAgI
CAgICAgICAgPENyeXB0b0luZm9ybWF0aW9uPgogICAgICAgICAgICAgICAgICAgIDxWYWxpZGF0aW9uT2JqZWN
0SWQgVk9SZWZlcmVuY2U9IlMtREUxQTdCMzI0OEY3MEJCNjNFODlEMUUwMjE4MzMwMzczRTI5MDA1QTg4NjcwR
UU5NzU4RkI1NEVDRUFCRTVENyIvPgogICAgICAgICAgICAgICAgICAgIDxBbGdvcml0aG0+aHR0cDovL3d3dy5

3My5vcmcvMjAwMS8wNC94bWxkc2lnLW1vcmUjcnNhLXNoYTI1NjwvQWxnb3JpdGhtPgogICAgICAgICAgICAgICAgI
CAgICAgIDxTZWN1cmVBbGdvcml0aG0+dHJ1ZTwvU2VjdXJlQWxnb3JpdGhtPgogICAgICAgICAgICAgICAgICAgICA
gIDxOb3RBZnRlcj4yMDIyLTEyLTMxVDIzOjAwOjAwWjwvTm90QWZ0ZXI+CiAgICAgICAgICAgICAgICAgICA8L0Nye
XB0b0luZm9ybWF0aW9uPgogICAgICAgICAgICAgICA8L0Fzc29jaWF0ZWRWYWxpZGF0aW9uUmVwb3J0RGF0YT4KICA
gICAgICA8L1NpZ25hdHVyZVZhbGlkYXRpb25TdGF0dXM+CiAgICA8L1NpZ25hdHVyZVZhbGlkYXRpb25SZXBvc
nQ+CiAgICA8U2lnbmF0dXJlVmFsaWRhdGlvbk9iamVjdHM+CiAgICAgICAgPFZhbGlkYXRpb25PYmplY3QgaWQ
9IkMtRjBGRjBCNDUxNEQzMTYzMDRGMjgxN0RCQTBCRkIwNURFREI5ODUyN0MwRTQ3QzczRThEOEZERkUxNkRGM
jY3RSI+CiAgICAgICAgICAgIDxPYmplY3RUeXBlPnVybjpldHNpOjAxOTEwMjp2YWxpZGF0aW9uT2JqZWN0OmN
lcnRpZmljYXRlPC9PYmplY3RUeXBlPgogICAgICAgICAgICA8VmFsaWRhdGlvbk9iamVjdFJlcHJlc2VudGF0a
W9uPgogICAgICAgICAgICAgICAgPERpZ2VzdEFsZ0FuZFZhbHVlPgogICAgICAgICAgICAgICAgICAgIDxuczI
6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhMjU2I
i8+CiAgICAgICAgICAgICAgICAgICAgPG5zMjpEaWdlc3RWYWx1ZT44UDhMUlJUVEZqQlBLQmZib0wrd1hlMjV
oU2ZBNUh4ejZOajkvaGJmSm40PTwvbnMyOkRpZ2VzdFZhbHVlPgogICAgICAgICAgICAgICAgPC9EaWdlc3RBb
GdBbmRWYWx1ZT4KICAgICAgICAgICAgPC9WYWxpZGF0aW9uT2JqZWN0UmVwcmVzZW50YXRpb24+CiAgICAgICA
gICAgIDxQT0U+CiAgICAgICAgICAgICAgICA8UE9FVGltZT4yMDE5LTA4LTIzVDA2OjA4OjQ1WjwvUE9FVGltZ
T4KICAgICAgICAgICAgICAgIDxUeXBlT2ZQcm9vZj51cm46ZXRzaTowMTkxMDI6cG9ldHlwZTp2YWxpZGF0aW9
uPC9UeXBlT2ZQcm9vZj4KICAgICAgICAgICAgPC9QT0U+CiAgICAgICAgPC9WYWxpZGF0aW9uT2JqZWN0PgogI
CAgICAgIDxWYWxpZGF0aW9uT2JqZWN0IGlkPSJDLUIyRUJFRDU1QzZDOTVCNzNBOUMyMjFMDVDMjkxMDBCNkY
1MjM0QzEzRDc4RTA4QjIxMjY1ODNDNDRjBGRjk2MUYiPgogICAgICAgICAgICA8T2JqZWN0VHlwZT51cm46ZXRza
TowMTkxMDI6dmFsaWRhdGlvbk9iamVjdDpjZXJ0aWZpY2F0ZTwvT2JqZWN0VHlwZT4KICAgICAgICAgICAgPFZ
hbGlkYXRpb25PYmplY3RSZXByZXNlbnRhdGlvbj4KICAgICAgICAgICAgICAgIDxEaWdlc3RBbGdBbmRWYWx1Z
T4KICAgICAgICAgICAgICAgICAgICA8bnMyOkRpZ2VzdE1ldGhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzM
ub3JnLzIwMDEvMDQveG1sZW5jI3NoYTI1NiIvPgogICAgICAgICAgICAgICAgICAgIDxuczI6RGlnZXN0VmFsd
WU+c3V2dFZjYkpXM09wd2lMZ1hDa1FDMjlTTk1FOWVPQ0xxJU1pZUFBEL2xoOD08L25zMjpEaWdlc3RWYWx1ZT4
KICAgICAgICAgICAgICAgIDwvRGlnZXN0QWxnQW5kVmFsdWU+CiAgICAgICAgICAgIDwvVmFsaWRhdGlvbk9ia
mVjdFJlcHJlc2VudGF0aW9uPgogICAgICAgICAgICA8UE9FPgogICAgICAgICAgICAgICAgPFBPRVRpbWU+MjA
xOS0wOC0yM1QwNjowODo0NVo8L1BPRVRpbWU+CiAgICAgICAgICAgICAgICA8VHlwZU9mUHJvb2Y+dXJuOmV0c
2k6MDE5MTAyOnBvZXR5cGU6dmFsaWRhdGlvbjwvVHlwZU9mUHJvb2Y+CiAgICAgICAgICAgIDwvUE9FPgogICA
gICAgIDwvVmFsaWRhdGlvbk9iamVjdD4KICAgICAgICA8VmFsaWRhdGlvbk9iamVjdCBpZD0iQy1DRENCNUQwM
0NEOUQ3MjY3NkQ1QzgyOUJGNTIyRUI3NEE3Qzc2NkE5N0I5RDVFQkEwOUE0NTNCNThEQjc0RDNFIj4KICAgICA
gICAgICAgPE9iamVjdFR5cGU+dXJuOmV0c2k6MDE5MTAyOnZhbGlkYXRpb25PYmplY3Q6Y2VydGlmaWNhdGU8L
09iamVjdFR5cGU+CiAgICAgICAgICAgIDxWYWxpZGF0aW9uT2JqZWN0UmVwcmVzZW50YXRpb24+CiAgICAgICA
gICAgICAgICA8RGlnZXN0QWxnQW5kVmFsdWU+CiAgICAgICAgICAgICAgICAgICAgPG5zMjpEaWdlc3RNZXRob
2QgQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVuYyNzaGEyNTYiLz4KICAgICAgICA
gICAgICAgICAgICA8bnMyOkRpZ2VzdFZhbHVlPnpjdGRBODJkY21kdFhJS2I5U0xyZEtmSFpxbDduVjY2Q2FSV
HRZMjNUVDQ9PC9uczI6RGlnZXN0VmFsdWU+CiAgICAgICAgICAgICAgICA8L0RpZ2VzdEFsZ0FuZFZhbHVlPgo
gICAgICAgICAgICA8L1ZhbGlkYXRpb25PYmplY3RSZXByZXNlbnRhdGlvbj4KICAgICAgICAgICAgPFBPRT4KI
CAgICAgICAgICAgICAgIDxQT0VUaW1lPjIwMTktMDgtMjNUMDY6MDg6NDVaPC9QT0VUaW1lPgogICAgICAgICA
gICAgICAgPFR5cGVPZlByb29mPnVybjpldHNpOjAxOTEwMjpwb2V0eXBlOnZhbGlkYXRpb248L1R5cGVPZlByb
29mPgogICAgICAgICAgICA8L1BPRT4KICAgICAgICA8L1ZhbGlkYXRpb25PYmplY3Q+CiAgICAgICAgPFZhbGl
kYXRpb25PYmplY3QgaWQ9IkMtNkYzNURFMzk2NUI5QTY5QkMzNjYyRDFBMzU1QjBBRTYwOTA3QURCNzQxQ0MxO
TExRUZEMEYzQkU3MkQ2QTZFOSI+CiAgICAgICAgICAgIDxPYmplY3RUeXBlPnVybjpldHNpOjAxOTEwMjp2YWx
pZGF0aW9uT2JqZWN0OmNlcnRpZmljYXRlPC9PYmplY3RUeXBlPgogICAgICAgICAgICA8VmFsaWRhdGlvbk9ia
mVjdFJlcHJlc2VudGF0aW9uPgogICAgICAgICAgICAgICAgPERpZ2VzdEFsZ0FuZFZhbHVlPgogICAgICAgICA
gICAgICAgICAgIDxuczI6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wN
C94bWxlbmMjc2hhMjU2Ii8+CiAgICAgICAgICAgICAgICAgICAgPG5zMjpEaWdlc3RWYWx1ZT5ielhlT1dXNXB
wdkRaaDBhTlZzSzVna0hyYmRCekJrUjc5RHp2bkxXcHVrPTwvbnMyOkRpZ2VzdFZhbHVlPgogICAgICAgICAgI
CAgICAgPC9EaWdlc3RBbGdBbmRWYWx1ZT4KICAgICAgICAgICAgPC9WYWxpZGF0aW9uT2JqZWN0UmVwcmVzZW5
0YXRpb24+CiAgICAgICAgICAgIDxQT0U+CiAgICAgICAgICAgICAgICA8UE9FVGltZT4yMDE5LTA4LTIzVDA2O
jA4OjQ1WjwvUE9FVGltZT4KICAgICAgICAgICAgICAgIDxUeXBlT2ZQcm9vZj51cm46ZXRzaTowMTkxMDI6cG9
ldHlwZTp2YWxpZGF0aW9uPC9UeXBlT2ZQcm9vZj4KICAgICAgICAgICAgPC9QT0U+CiAgICAgICAgPC9WYWxpZ

```
GF0aW9uT2JqZWN0PgogICAgICAgIDxWYWxpZGF0aW9uT2JqZWN0IGlkPSJELUM1OEM4MEE4MDUzMEUwRjM0OUJ
DMzJERUY1MDI4MEQ3NEM0QjdFQkYyNTI4MDQ0MDE4MTE2N0EyRjFBMEIzMUQiPgogICAgICAgICA8T2JqZ
WN0VHlwZT51cm46ZXRzaTowMTkxMDI6dmFsaWRhdGlvbk9iamVjdDpzaWduZWREYXRhPC9PYmplY3RUeXBlPgo
gICAgICAgICA8VmFsaWRhdGlvbk9iamVjdFJlcHJlc2VudGF0aW9uPgogICAgICAgICAgICAgICAgPERpZ
2VzdEFsZ0FuZFZhbHVlPgogICAgICAgICAgICAgICAgICAgIDxuczI6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0
iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhMjU2Ii8+CiAgICAgICAgICAgICAgICAgICAgP
G5zMjpEaWdlc3RWYWx1ZT5rY0RIT1pqd1poVmZ1RGh1aENlQ0VSUm1ZcFRINEpqNFJtZlZWaTMxUTlnPTwvbnM
yOkRpZ2VzdFZhbHVlPgogICAgICAgICAgICAgICAgPC9EaWdlc3RBbGdBbmRWYWx1ZT4KICAgICAgICAgICAgP
C9WYWxpZGF0aW9uT2JqZWN0UmVwcmVzZW50YXRpb24+CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICA8UE9FVGltZT4yMDE5LTA4LTIzVDA2OjA4OjQ1WjwvUE9FVGltZT4KICAgICAgICAgICAgICAgICAgICAgICAgIDxUeXBlT
2ZQcm9vZj51cm46ZXRzaTowMTkxMDI6cG9ldHlwZTp2YWxpZGF0aW9uPC9UeXBlT2ZQcm9vZj4KICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgPC9WYWxpZGF0aW9uT2JqZWN0PgogICAgICAgICAgICAgICAgICAgICAgICAgICAgPC9TaWduYXR1cmVWYWxpZGF0aW9uT
2JqZWN0cz4KPC9WYWxpZGF0aW9uUmVwb3J0Pgo=",
  "DiagnosticData" : {
    "DocumentName" : "xades-detached.xml",
    "ValidationDate" : "2019-08-23T06:08:45",
    "ContainerInfo" : null,
    "Signature" : [ {
      "Id" : "S-DE1A7B3248F70BB63E89D1E0218330373E29005A88670EE9758FB54ECEABE5D7",
      "DAIdentifier" : "id-afde782436468dd74eeb181f7ce110e1",
      "SignatureFilename" : "xades-detached.xml",
      "ErrorMessage" : null,
      "DateTime" : "2017-09-28T11:09:04",
      "SignatureFormat" : "XAdES-BASELINE-B",
      "StructuralValidation" : {
        "Valid" : true,
        "Message" : null
      },
      "DigestMatcher" : [ {
        "DataFound" : true,
        "DataIntact" : true,
        "DigestMethod" : "SHA256",
        "DigestValue" : "kcDHOZjwZhVfuDhuhCeCERRmYpTH4Jj4RmfVVi31Q9g=",
        "type" : "REFERENCE",
        "name" : "r-id-1"
      }, {
        "DataFound" : true,
        "DataIntact" : true,
        "DigestMethod" : "SHA256",
        "DigestValue" : "DztwNTmRoOAm6/lMI8Rym5xZPzIvLYDzn/ebYYkPsr4=",
        "type" : "SIGNED_PROPERTIES",
        "name" : "#xades-id-afde782436468dd74eeb181f7ce110e1"
      } ],
      "BasicSignature" : {
        "EncryptionAlgoUsedToSignThisToken" : "RSA",
        "KeyLengthUsedToSignThisToken" : "2048",
        "DigestAlgoUsedToSignThisToken" : "SHA256",
        "MaskGenerationFunctionUsedToSignThisToken" : null,
        "SignatureIntact" : true,
        "SignatureValid" : true
      },
```

```
      "SigningCertificate" : {
        "AttributePresent" : true,
        "DigestValuePresent" : true,
        "DigestValueMatch" : true,
        "IssuerSerialMatch" : true,
        "Certificate" : "C-
F0FF0B4514D316304F2817DBA0BFB05DEDB98527C0E47C73E8D8FDFE16DF267E"
      },
      "ChainItem" : [ {
        "Certificate" : "C-
F0FF0B4514D316304F2817DBA0BFB05DEDB98527C0E47C73E8D8FDFE16DF267E"
      }, {
        "Certificate" : "C-
6F35DE3965B9A69BC3661D1A355B0AE60907ADB741CC1911EFD0F3BE72D6A6E9"
      } ],
      "ContentType" : null,
      "MimeType" : "text/xml",
      "ContentIdentifier" : null,
      "ContentHints" : null,
      "SignatureProductionPlace" : null,
      "Indication" : [ ],
      "SignerRole" : [ ],
      "Policy" : null,
      "PDFSignatureDictionary" : null,
      "SignerDocumentRepresentations" : {
        "HashOnly" : false,
        "DocHashOnly" : false
      },
      "FoundCertificates" : {
        "RelatedCertificate" : [ {
          "Origin" : [ "KEY_INFO" ],
          "CertificateRef" : [ {
            "Origin" : "SIGNING_CERTIFICATE",
            "IssuerSerial" :
"MFYwUaRPME0xEDAOBgNVBAMMB2dvb2QtY2ExGTAXBgNVBAoMEE5vd2luYSBTb2x1dGlvbnMxETAPBgNVBAsMC
FBLSS1URVNUMQswCQYDVQQGEwJMVQIBCg==",
            "DigestAlgoAndValue" : {
              "DigestMethod" : "SHA1",
              "DigestValue" : "c+Vohg0jIcZ4UQSWeglCg0oGNWs="
            }
          } ],
          "Certificate" : "C-
F0FF0B4514D316304F2817DBA0BFB05DEDB98527C0E47C73E8D8FDFE16DF267E"
        }, {
          "Origin" : [ "KEY_INFO" ],
          "CertificateRef" : [ ],
          "Certificate" : "C-
6F35DE3965B9A69BC3661D1A355B0AE60907ADB741CC1911EFD0F3BE72D6A6E9"
        } ],
        "OrphanCertificate" : [ ]
      },
```

```
      "FoundRevocations" : {
        "RelatedRevocation" : [ ],
        "OrphanRevocation" : [ ]
      },
      "FoundTimestamp" : [ ],
      "SignatureScope" : [ {
        "Scope" : "FULL",
        "Name" : "sample.xml",
        "Description" : "Full document",
        "Transformation" : null,
        "SignerData" : "D-
C58C80A80530E0F349BC32DEF50280D74C4B7EBF25280440181167A2F1A0B31D"
      } ],
      "SignatureDigestReference" : {
        "CanonicalizationMethod" : "http://www.w3.org/2001/10/xml-exc-c14n#",
        "DigestMethod" : "SHA256",
        "DigestValue" : "SXLcmUDMsYRI6Fz6pek8zrxrZbkyyZOIFVzmJJuWPm4="
      },
      "SignatureValue" :
"YA7sENt3N8ufLFMnKr36r0PqzMiY3Q0s++IGTEUC0spaxUv0dHZM0d/yn3kpLJLoUkI4M3flj5WGn83kf05Bq
M1khsX61GJzaFTPGpm7akRQKhvoH25yyqTYXESlBcm04iziKhLMzZjUfx4/B1ZIysv5pIBgJ2r2oi6jLop9ww3
ge4c4YJoaK+SXk6hyTNOcN8PjGe63WYOTNVPQFvja8Bnwg+a0bBuwD+8N6fwigCdW5a/4DJUe/J8Mb70ZI8PoO
znGDfi+TPbiIeVmCbl5mUoUg2Q/xYluJfLh3uGQAXKBvF45oDIHRVefnN/D/WytAClUVDoQSywemnjPpqF8eg=
=",
      "CounterSignature" : null,
      "Parent" : null
    } ],
    "Certificate" : [ {
      "Id" : "C-F0FF0B4514D316304F2817DBA0BFB05DEDB98527C0E47C73E8D8FDFE16DF267E",
      "SubjectDistinguishedName" : [ {
        "value" : "c=lu,ou=pki-test,o=nowina solutions,cn=good-user",
        "Format" : "CANONICAL"
      }, {
        "value" : "C=LU,OU=PKI-TEST,O=Nowina Solutions,CN=good-user",
        "Format" : "RFC2253"
      } ],
      "IssuerDistinguishedName" : [ {
        "value" : "c=lu,ou=pki-test,o=nowina solutions,cn=good-ca",
        "Format" : "CANONICAL"
      }, {
        "value" : "C=LU,OU=PKI-TEST,O=Nowina Solutions,CN=good-ca",
        "Format" : "RFC2253"
      } ],
      "SerialNumber" : 10,
      "CommonName" : "good-user",
      "Locality" : null,
      "State" : null,
      "CountryName" : "LU",
      "OrganizationName" : "Nowina Solutions",
      "GivenName" : null,
      "OrganizationalUnit" : "PKI-TEST",
```

```
      "Surname" : null,
      "Pseudonym" : null,
      "Email" : null,
      "aiaUrl" : [ "http://dss.nowina.lu/pki-factory/crt/good-ca.crt" ],
      "crlUrl" : [ ],
      "ocspServerUrl" : [ "http://dss.nowina.lu/pki-factory/ocsp/good-ca" ],
      "Source" : [ "SIGNATURE" ],
      "NotAfter" : "2018-08-26T07:54:31",
      "NotBefore" : "2016-10-26T07:54:31",
      "PublicKeySize" : 2048,
      "PublicKeyEncryptionAlgo" : "RSA",
      "KeyUsage" : [ "nonRepudiation" ],
      "extendedKeyUsagesOid" : [ ],
      "IdPkixOcspNoCheck" : false,
      "BasicSignature" : {
        "EncryptionAlgoUsedToSignThisToken" : "RSA",
        "KeyLengthUsedToSignThisToken" : "2048",
        "DigestAlgoUsedToSignThisToken" : "SHA256",
        "MaskGenerationFunctionUsedToSignThisToken" : null,
        "SignatureIntact" : true,
        "SignatureValid" : true
      },
      "SigningCertificate" : {
        "AttributePresent" : null,
        "DigestValuePresent" : null,
        "DigestValueMatch" : null,
        "IssuerSerialMatch" : null,
        "Certificate" : "C-
6F35DE3965B9A69BC3661D1A355B0AE60907ADB741CC1911EFD0F3BE72D6A6E9"
      },
      "ChainItem" : [ {
        "Certificate" : "C-
6F35DE3965B9A69BC3661D1A355B0AE60907ADB741CC1911EFD0F3BE72D6A6E9"
      } ],
      "Trusted" : false,
      "SelfSigned" : false,
      "certificatePolicy" : [ ],
      "qcStatementOid" : [ ],
      "qcTypeOid" : [ ],
      "TrustedServiceProvider" : [ ],
      "CertificateRevocation" : [ ],
      "Base64Encoded" : null,
      "DigestAlgoAndValue" : {
        "DigestMethod" : "SHA256",
        "DigestValue" : "8P8LRRTTFjBPKBfboL+wXe25hSfA5Hxz6Nj9/hbfJn4="
      }
    }, {
      "Id" : "C-B2EBED55C6C95B73A9C222E05C29100B6F5234C13D78E08B2126583CF0FF961F",
      "SubjectDistinguishedName" : [ {
        "value" : "c=lu,ou=pki-test,o=nowina solutions,cn=good-ca",
        "Format" : "CANONICAL"
```

```
      }, {
        "value" : "C=LU,OU=PKI-TEST,O=Nowina Solutions,CN=good-ca",
        "Format" : "RFC2253"
      } ],
      "IssuerDistinguishedName" : [ {
        "value" : "c=lu,ou=pki-test,o=nowina solutions,cn=root-ca",
        "Format" : "CANONICAL"
      }, {
        "value" : "C=LU,OU=PKI-TEST,O=Nowina Solutions,CN=root-ca",
        "Format" : "RFC2253"
      } ],
      "SerialNumber" : 4,
      "CommonName" : "good-ca",
      "Locality" : null,
      "State" : null,
      "CountryName" : "LU",
      "OrganizationName" : "Nowina Solutions",
      "GivenName" : null,
      "OrganizationalUnit" : "PKI-TEST",
      "Surname" : null,
      "Pseudonym" : null,
      "Email" : null,
      "aiaUrl" : [ "http://dss.nowina.lu/pki-factory/crt/root-ca.crt" ],
      "crlUrl" : [ "http://dss.nowina.lu/pki-factory/crl/root-ca.crl" ],
      "ocspServerUrl" : [ ],
      "Source" : [ "AIA" ],
      "NotAfter" : "2020-07-20T12:31:08",
      "NotBefore" : "2018-09-20T12:31:08",
      "PublicKeySize" : 2048,
      "PublicKeyEncryptionAlgo" : "RSA",
      "KeyUsage" : [ "keyCertSign", "crlSign" ],
      "extendedKeyUsagesOid" : [ ],
      "IdPkixOcspNoCheck" : false,
      "BasicSignature" : {
        "EncryptionAlgoUsedToSignThisToken" : "RSA",
        "KeyLengthUsedToSignThisToken" : "2048",
        "DigestAlgoUsedToSignThisToken" : "SHA256",
        "MaskGenerationFunctionUsedToSignThisToken" : null,
        "SignatureIntact" : true,
        "SignatureValid" : true
      },
      "SigningCertificate" : {
        "AttributePresent" : null,
        "DigestValuePresent" : null,
        "DigestValueMatch" : null,
        "IssuerSerialMatch" : null,
        "Certificate" : "C-
CDCB5D03CD9D72676D5C829BF522EB74A7C766A97B9D5EBA09A453B58DB74D3E"
      },
      "ChainItem" : [ {
        "Certificate" : "C-
```

```
CDCB5D03CD9D72676D5C829BF522EB74A7C766A97B9D5EBA09A453B58DB74D3E"
      } ],
      "Trusted" : false,
      "SelfSigned" : false,
      "certificatePolicy" : [ ],
      "qcStatementOid" : [ ],
      "qcTypeOid" : [ ],
      "TrustedServiceProvider" : [ ],
      "CertificateRevocation" : [ ],
      "Base64Encoded" : null,
      "DigestAlgoAndValue" : {
        "DigestMethod" : "SHA256",
        "DigestValue" : "suvtVcbJW3OpwiLgXCkQC29SNME9eOCLISZYPPD/lh8="
      }
    }, {
      "Id" : "C-CDCB5D03CD9D72676D5C829BF522EB74A7C766A97B9D5EBA09A453B58DB74D3E",
      "SubjectDistinguishedName" : [ {
        "value" : "c=lu,ou=pki-test,o=nowina solutions,cn=root-ca",
        "Format" : "CANONICAL"
      }, {
        "value" : "C=LU,OU=PKI-TEST,O=Nowina Solutions,CN=root-ca",
        "Format" : "RFC2253"
      } ],
      "IssuerDistinguishedName" : [ {
        "value" : "c=lu,ou=pki-test,o=nowina solutions,cn=root-ca",
        "Format" : "CANONICAL"
      }, {
        "value" : "C=LU,OU=PKI-TEST,O=Nowina Solutions,CN=root-ca",
        "Format" : "RFC2253"
      } ],
      "SerialNumber" : 1,
      "CommonName" : "root-ca",
      "Locality" : null,
      "State" : null,
      "CountryName" : "LU",
      "OrganizationName" : "Nowina Solutions",
      "GivenName" : null,
      "OrganizationalUnit" : "PKI-TEST",
      "Surname" : null,
      "Pseudonym" : null,
      "Email" : null,
      "aiaUrl" : [ ],
      "crlUrl" : [ ],
      "ocspServerUrl" : [ ],
      "Source" : [ "AIA" ],
      "NotAfter" : "2020-08-20T12:31:07",
      "NotBefore" : "2018-08-20T12:31:07",
      "PublicKeySize" : 2048,
      "PublicKeyEncryptionAlgo" : "RSA",
      "KeyUsage" : [ "keyCertSign", "crlSign" ],
      "extendedKeyUsagesOid" : [ ],
```

```
      "IdPkixOcspNoCheck" : false,
      "BasicSignature" : {
        "EncryptionAlgoUsedToSignThisToken" : "RSA",
        "KeyLengthUsedToSignThisToken" : "2048",
        "DigestAlgoUsedToSignThisToken" : "SHA512",
        "MaskGenerationFunctionUsedToSignThisToken" : null,
        "SignatureIntact" : true,
        "SignatureValid" : true
      },
      "SigningCertificate" : null,
      "ChainItem" : null,
      "Trusted" : false,
      "SelfSigned" : true,
      "certificatePolicy" : [ ],
      "qcStatementOid" : [ ],
      "qcTypeOid" : [ ],
      "TrustedServiceProvider" : [ ],
      "CertificateRevocation" : [ ],
      "Base64Encoded" : null,
      "DigestAlgoAndValue" : {
        "DigestMethod" : "SHA256",
        "DigestValue" : "zctdA82dcmdtXIKb9SLrdKfHZql7nV66CaRTtY23TT4="
      }
    }, {
      "Id" : "C-6F35DE3965B9A69BC3661D1A355B0AE60907ADB741CC1911EFD0F3BE72D6A6E9",
      "SubjectDistinguishedName" : [ {
        "value" : "c=lu,ou=pki-test,o=nowina solutions,cn=good-ca",
        "Format" : "CANONICAL"
      }, {
        "value" : "C=LU,OU=PKI-TEST,O=Nowina Solutions,CN=good-ca",
        "Format" : "RFC2253"
      } ],
      "IssuerDistinguishedName" : [ {
        "value" : "c=lu,ou=pki-test,o=nowina solutions,cn=root-ca",
        "Format" : "CANONICAL"
      }, {
        "value" : "C=LU,OU=PKI-TEST,O=Nowina Solutions,CN=root-ca",
        "Format" : "RFC2253"
      } ],
      "SerialNumber" : 4,
      "CommonName" : "good-ca",
      "Locality" : null,
      "State" : null,
      "CountryName" : "LU",
      "OrganizationName" : "Nowina Solutions",
      "GivenName" : null,
      "OrganizationalUnit" : "PKI-TEST",
      "Surname" : null,
      "Pseudonym" : null,
      "Email" : null,
      "aiaUrl" : [ "http://dss.nowina.lu/pki-factory/crt/root-ca.crt" ],
```

```
      "crlUrl" : [ "http://dss.nowina.lu/pki-factory/crl/root-ca.crl" ],
      "ocspServerUrl" : [ ],
      "Source" : [ "SIGNATURE" ],
      "NotAfter" : "2018-08-26T07:54:30",
      "NotBefore" : "2016-10-26T07:54:30",
      "PublicKeySize" : 2048,
      "PublicKeyEncryptionAlgo" : "RSA",
      "KeyUsage" : [ "digitalSignature" ],
      "extendedKeyUsagesOid" : [ ],
      "IdPkixOcspNoCheck" : false,
      "BasicSignature" : {
        "EncryptionAlgoUsedToSignThisToken" : "RSA",
        "KeyLengthUsedToSignThisToken" : "?",
        "DigestAlgoUsedToSignThisToken" : "SHA256",
        "MaskGenerationFunctionUsedToSignThisToken" : null,
        "SignatureIntact" : false,
        "SignatureValid" : false
      },
      "SigningCertificate" : null,
      "ChainItem" : null,
      "Trusted" : false,
      "SelfSigned" : false,
      "certificatePolicy" : [ ],
      "qcStatementOid" : [ ],
      "qcTypeOid" : [ ],
      "TrustedServiceProvider" : [ ],
      "CertificateRevocation" : [ ],
      "Base64Encoded" : null,
      "DigestAlgoAndValue" : {
        "DigestMethod" : "SHA256",
        "DigestValue" : "bzXeOWW5ppvDZh0aNVsK5gkHrbdBzBkR79DzvnLWpuk="
      }
    } ],
    "Revocation" : [ ],
    "Timestamp" : [ ],
    "OrphanToken" : [ ],
    "SignerData" : [ {
      "Id" : "D-C58C80A80530E0F349BC32DEF50280D74C4B7EBF25280440181167A2F1A0B31D",
      "ReferencedName" : "sample.xml",
      "DigestAlgoAndValue" : {
        "DigestMethod" : "SHA256",
        "DigestValue" : "kcDHOZjwZhVfuDhuhCeCERRmYpTH4Jj4RmfVVi31Q9g="
      }
    } ],
    "TrustedList" : [ ],
    "ListOfTrustedLists" : null
  },
  "SimpleReport" : {
    "Policy" : {
      "PolicyName" : "QES AdESQC TL based",
      "PolicyDescription" : "Validate electronic signatures and indicates whether they
```

```
are Advanced electronic Signatures (AdES), AdES supported by a Qualified Certificate
(AdES/QC) or a\n\t\tQualified electronic Signature (QES). All certificates and their
related chains supporting the signatures are validated against the EU Member State
Trusted Lists (this includes\n\t\tsigner's certificate and certificates used to
validate certificate validity status services - CRLs, OCSP, and time-stamps).\n\t"
    },
    "ValidationTime" : "2019-08-23T06:08:45",
    "DocumentName" : "xades-detached.xml",
    "ValidSignaturesCount" : 0,
    "SignaturesCount" : 1,
    "ContainerType" : null,
    "Signature" : [ {
      "Filename" : null,
      "SigningTime" : "2017-09-28T11:09:04",
      "BestSignatureTime" : "2019-08-23T06:08:45",
      "SignedBy" : "C-
F0FF0B4514D316304F2817DBA0BFB05DEDB98527C0E47C73E8D8FDFE16DF267E",
      "CertificateChain" : {
        "Certificate" : [ {
          "id" : "C-F0FF0B4514D316304F2817DBA0BFB05DEDB98527C0E47C73E8D8FDFE16DF267E",
          "qualifiedName" : "good-user"
        }, {
          "id" : "C-6F35DE3965B9A69BC3661D1A355B0AE60907ADB741CC1911EFD0F3BE72D6A6E9",
          "qualifiedName" : "good-ca"
        } ]
      },
      "SignatureLevel" : {
        "value" : "N/A",
        "description" : "Not applicable"
      },
      "Indication" : "INDETERMINATE",
      "SubIndication" : "NO_CERTIFICATE_CHAIN_FOUND",
      "Errors" : [ "Unable to build a certificate chain until a trusted list!", "The
result of the LTV validation process is not acceptable to continue the process!" ],
      "Warnings" : [ "The signature/seal is an INDETERMINATE AdES!" ],
      "Infos" : [ ],
      "SignatureScope" : [ {
        "value" : "Full document",
        "name" : "sample.xml",
        "scope" : "FULL"
      } ],
      "Id" : "S-DE1A7B3248F70BB63E89D1E0218330373E29005A88670EE9758FB54ECEABE5D7",
      "CounterSignature" : null,
      "ParentId" : null,
      "SignatureFormat" : "XAdES-BASELINE-B"
    } ]
  },
  "DetailedReport" : {
    "Signatures" : [ {
      "ValidationProcessBasicSignatures" : {
        "Constraint" : [ {
```

```
          "Name" : {
            "value" : "Is the result of the Basic Validation Process conclusive?",
            "NameId" : "ADEST_ROBVPIIC"
          },
          "Status" : "NOT OK",
          "Error" : {
            "value" : "The result of the Basic validation process is not conclusive!",
            "NameId" : "ADEST_ROBVPIIC_ANS"
          },
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : "S-DE1A7B3248F70BB63E89D1E0218330373E29005A88670EE9758FB54ECEABE5D7"
        } ],
        "Conclusion" : {
          "Indication" : "INDETERMINATE",
          "SubIndication" : "NO_CERTIFICATE_CHAIN_FOUND",
          "Errors" : [ {
            "value" : "The certificate chain for signature is not trusted, there is no
trusted anchor.",
            "NameId" : "BBB_XCV_CCCBB_SIG_ANS"
          } ],
          "Warnings" : null,
          "Infos" : null
        },
        "Title" : "Validation Process for Basic Signatures",
        "ProofOfExistence" : {
          "Time" : "2019-08-23T06:08:45",
          "TimestampId" : null
        }
      },
      "ValidationProcessTimestamps" : [ ],
      "ValidationProcessLongTermData" : {
        "Constraint" : [ {
          "Name" : {
            "value" : "Is the result of the Basic Validation Process acceptable?",
            "NameId" : "LTV_ABSV"
          },
          "Status" : "NOT OK",
          "Error" : {
            "value" : "The result of the Basic validation process is not acceptable to
continue the process!",
            "NameId" : "LTV_ABSV_ANS"
          },
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        } ],
        "Conclusion" : {
          "Indication" : "INDETERMINATE",
```

```
          "SubIndication" : "NO_CERTIFICATE_CHAIN_FOUND",
          "Errors" : [ {
            "value" : "The certificate chain for signature is not trusted, there is no
trusted anchor.",
            "NameId" : "BBB_XCV_CCCBB_SIG_ANS"
          } ],
          "Warnings" : null,
          "Infos" : null
        },
        "Title" : "Validation Process for Signatures with Time and Signatures with
Long-Term Validation Data",
        "ProofOfExistence" : {
          "Time" : "2019-08-23T06:08:45",
          "TimestampId" : null
        }
      },
      "ValidationProcessArchivalData" : {
        "Constraint" : [ {
          "Name" : {
            "value" : "Is the result of the LTV validation process acceptable?",
            "NameId" : "ARCH_LTVV"
          },
          "Status" : "NOT OK",
          "Error" : {
            "value" : "The result of the LTV validation process is not acceptable to
continue the process!",
            "NameId" : "ARCH_LTVV_ANS"
          },
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        } ],
        "Conclusion" : {
          "Indication" : "INDETERMINATE",
          "SubIndication" : "NO_CERTIFICATE_CHAIN_FOUND",
          "Errors" : [ {
            "value" : "The certificate chain for signature is not trusted, there is no
trusted anchor.",
            "NameId" : "BBB_XCV_CCCBB_SIG_ANS"
          } ],
          "Warnings" : null,
          "Infos" : null
        },
        "Title" : "Validation Process for Signatures with Archival Data",
        "ProofOfExistence" : {
          "Time" : "2019-08-23T06:08:45",
          "TimestampId" : null
        }
      },
      "ValidationSignatureQualification" : {
```

```
        "ValidationCertificateQualification" : [ ],
        "Constraint" : [ {
          "Name" : {
            "value" : "Is the signature/seal an acceptable AdES (ETSI EN 319 102-1)
?",
            "NameId" : "QUAL_IS_ADES"
          },
          "Status" : "WARNING",
          "Error" : null,
          "Warning" : {
            "value" : "The signature/seal is an INDETERMINATE AdES!",
            "NameId" : "QUAL_IS_ADES_IND"
          },
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        }, {
          "Name" : {
            "value" : "Has a trusted list been reached for the certificate chain?",
            "NameId" : "QUAL_CERT_TRUSTED_LIST_REACHED"
          },
          "Status" : "NOT OK",
          "Error" : {
            "value" : "Unable to build a certificate chain until a trusted list!",
            "NameId" : "QUAL_CERT_TRUSTED_LIST_REACHED_ANS"
          },
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        } ],
        "Conclusion" : {
          "Indication" : "FAILED",
          "SubIndication" : null,
          "Errors" : [ {
            "value" : "Unable to build a certificate chain until a trusted list!",
            "NameId" : "QUAL_CERT_TRUSTED_LIST_REACHED_ANS"
          }, {
            "value" : "Unable to build a certificate chain until a trusted list!",
            "NameId" : "QUAL_CERT_TRUSTED_LIST_REACHED_ANS"
          } ],
          "Warnings" : [ {
            "value" : "The signature/seal is an INDETERMINATE AdES!",
            "NameId" : "QUAL_IS_ADES_IND"
          } ],
          "Infos" : null
        },
        "Title" : "Signature Qualification",
        "Id" : "S-DE1A7B3248F70BB63E89D1E0218330373E29005A88670EE9758FB54ECEABE5D7",
        "SignatureQualification" : "N/A"
      },
```

```
      "Id" : "S-DE1A7B3248F70BB63E89D1E0218330373E29005A88670EE9758FB54ECEABE5D7",
      "CounterSignature" : null
    } ],
    "Certificate" : null,
    "BasicBuildingBlocks" : [ {
      "FC" : {
        "Constraint" : [ {
          "Name" : {
            "value" : "Is the expected format found?",
            "NameId" : "BBB_FC_IEFF"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        } ],
        "Conclusion" : {
          "Indication" : "PASSED",
          "SubIndication" : null,
          "Errors" : null,
          "Warnings" : null,
          "Infos" : null
        },
        "Title" : "Format Checking"
      },
      "ISC" : {
        "CertificateChain" : {
          "ChainItem" : [ {
            "Source" : "SIGNATURE",
            "Id" : "C-
F0FF0B4514D316304F2817DBA0BFB05DEDB98527C0E47C73E8D8FDFE16DF267E"
          }, {
            "Source" : "SIGNATURE",
            "Id" : "C-
6F35DE3965B9A69BC3661D1A355B0AE60907ADB741CC1911EFD0F3BE72D6A6E9"
          } ]
        },
        "Constraint" : [ {
          "Name" : {
            "value" : "Is there an identified candidate for the signing certificate?",
            "NameId" : "BBB_ICS_ISCI"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        }, {
```

```
          "Name" : {
            "value" : "Is the signed attribute: 'signing-certificate' present?",
            "NameId" : "BBB_ICS_ISASCP"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        }, {
          "Name" : {
            "value" : "Is the signed attribute: 'cert-digest' of the certificate
present?",
            "NameId" : "BBB_ICS_ISACDP"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        }, {
          "Name" : {
            "value" : "Is the certificate's digest value valid?",
            "NameId" : "BBB_ICS_ICDVV"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        }, {
          "Name" : {
            "value" : "Are the issuer distinguished name and the serial number
equal?",
            "NameId" : "BBB_ICS_AIDNASNE"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        } ],
        "Conclusion" : {
          "Indication" : "PASSED",
          "SubIndication" : null,
          "Errors" : null,
          "Warnings" : null,
```

```
          "Infos" : null
        },
        "Title" : "Identification of the Signing Certificate"
      },
      "VCI" : {
        "Constraint" : [ {
          "Name" : {
            "value" : "Is the signature policy known?",
            "NameId" : "BBB_VCI_ISPK"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        } ],
        "Conclusion" : {
          "Indication" : "PASSED",
          "SubIndication" : null,
          "Errors" : null,
          "Warnings" : null,
          "Infos" : null
        },
        "Title" : "Validation Context Initialization"
      },
      "XCV" : {
        "SubXCV" : [ ],
        "Constraint" : [ {
          "Name" : {
            "value" : "Can the certificate chain be built till the trust anchor?",
            "NameId" : "BBB_XCV_CCCBB"
          },
          "Status" : "NOT OK",
          "Error" : {
            "value" : "The certificate chain for signature is not trusted, there is no
trusted anchor.",
            "NameId" : "BBB_XCV_CCCBB_SIG_ANS"
          },
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        } ],
        "Conclusion" : {
          "Indication" : "INDETERMINATE",
          "SubIndication" : "NO_CERTIFICATE_CHAIN_FOUND",
          "Errors" : [ {
            "value" : "The certificate chain for signature is not trusted, there is no
trusted anchor.",
            "NameId" : "BBB_XCV_CCCBB_SIG_ANS"
```

```
        } ],
        "Warnings" : null,
        "Infos" : null
      },
      "Title" : "X509 Certificate Validation"
    },
    "CV" : {
      "Constraint" : [ {
        "Name" : {
          "value" : "Is the reference data object found?",
          "NameId" : "BBB_CV_IRDOF"
        },
        "Status" : "OK",
        "Error" : null,
        "Warning" : null,
        "Info" : null,
        "AdditionalInfo" : "Reference : r-id-1",
        "Id" : null
      }, {
        "Name" : {
          "value" : "Is the reference data object intact?",
          "NameId" : "BBB_CV_IRDOI"
        },
        "Status" : "OK",
        "Error" : null,
        "Warning" : null,
        "Info" : null,
        "AdditionalInfo" : "Reference : r-id-1",
        "Id" : null
      }, {
        "Name" : {
          "value" : "Is the reference data object found?",
          "NameId" : "BBB_CV_IRDOF"
        },
        "Status" : "OK",
        "Error" : null,
        "Warning" : null,
        "Info" : null,
        "AdditionalInfo" : "Reference : #xades-id-afde782436468dd74eeb181f7ce110e1",
        "Id" : null
      }, {
        "Name" : {
          "value" : "Is the reference data object intact?",
          "NameId" : "BBB_CV_IRDOI"
        },
        "Status" : "OK",
        "Error" : null,
        "Warning" : null,
        "Info" : null,
        "AdditionalInfo" : "Reference : #xades-id-afde782436468dd74eeb181f7ce110e1",
        "Id" : null
```

```
        }, {
          "Name" : {
            "value" : "Is the signature intact?",
            "NameId" : "BBB_CV_ISI"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        } ],
        "Conclusion" : {
          "Indication" : "PASSED",
          "SubIndication" : null,
          "Errors" : null,
          "Warnings" : null,
          "Infos" : null
        },
        "Title" : "Cryptographic Verification"
      },
      "SAV" : {
        "CryptographicInfo" : {
          "Algorithm" : "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
          "KeyLength" : "2048",
          "Secure" : true,
          "NotAfter" : "2022-12-31T23:00:00"
        },
        "Constraint" : [ {
          "Name" : {
            "value" : "Is signed qualifying property: 'signing-time' present?",
            "NameId" : "BBB_SAV_ISQPSTP"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        }, {
          "Name" : {
            "value" : "Is signed qualifying property: 'message-digest' or
'SignedProperties' present?",
            "NameId" : "BBB_SAV_ISQPMDOSPP"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
```

```
        }, {
          "Name" : {
            "value" : "Are signature cryptographic constraints met?",
            "NameId" : "ASCCM"
          },
          "Status" : "OK",
          "Error" : null,
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : "Validation time : 2019-08-23 06:08 for token with ID :
[S-DE1A7B3248F70BB63E89D1E0218330373E29005A88670EE9758FB54ECEABE5D7]",
          "Id" : null
        } ],
        "Conclusion" : {
          "Indication" : "PASSED",
          "SubIndication" : null,
          "Errors" : null,
          "Warnings" : null,
          "Infos" : null
        },
        "Title" : "Signature Acceptance Validation",
        "ValidationTime" : "2019-08-23T06:08:45"
      },
      "PSV" : null,
      "PCV" : null,
      "VTS" : null,
      "CertificateChain" : {
        "ChainItem" : [ {
          "Source" : "SIGNATURE",
          "Id" : "C-F0FF0B4514D316304F2817DBA0BFB05DEDB98527C0E47C73E8D8FDFE16DF267E"
        }, {
          "Source" : "SIGNATURE",
          "Id" : "C-6F35DE3965B9A69BC3661D1A355B0AE60907ADB741CC1911EFD0F3BE72D6A6E9"
        } ]
      },
      "Conclusion" : {
        "Indication" : "INDETERMINATE",
        "SubIndication" : "NO_CERTIFICATE_CHAIN_FOUND",
        "Errors" : [ {
          "value" : "The certificate chain for signature is not trusted, there is no
trusted anchor.",
          "NameId" : "BBB_XCV_CCCBB_SIG_ANS"
        } ],
        "Warnings" : null,
        "Infos" : null
      },
      "Id" : "S-DE1A7B3248F70BB63E89D1E0218330373E29005A88670EE9758FB54ECEABE5D7",
      "Type" : "SIGNATURE"
    } ],
    "TLAnalysis" : [ ]
  }
```

```
}
```

Retrieve original document(s)

This service returns the signed data for a given signature.

Request

```
POST /services/rest/validation/getOriginalDocuments HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 8947

{
  "signedDocument" : {
    "bytes" :
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48ZHM6U2lnbmF0dXJlIHhtbG5zOmRzPSJod
HRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjIiBJZD0iaWQtZWExMGEwNTE3Y2JjN2Y1NDllYTNlNjg
1ODY3YWM5NWUiPjxkczpTaWduZWRJbmZvPjxkczpDYW5vbmljYWxpemF0aW9uTWV0aG9kIEFsZ29yaXRobT0ia
HR0cDovL3d3dy53My5vcmcvVFIvMjAwMS9SRUMteG1sLWMxNG4tMjAwMTAzMTUiLz48ZHM6U2lnbmF0dXJlTWV
0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxkc2lnLW1vcmUjcnNhLXNoYTI1N
iIvPjxkczpSZWZlcmVuY2UgSWQ9InItaWQtMSIgVHlwZT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWx
kc2lnI09iamVjdCIgVVJJPSIjby1pZC0xIj48ZHM6VHJhbnNmb3Jtcz48ZHM6VHJhbnNmb3JtIEFsZ29yaXRob
T0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2Jhc2U2NCIvPjwvZHM6VHJhbnNmb3Jtcz48ZHM
6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhMjU2Ii
i8+PGRzOkRpZ2VzdFZhbHVlPkxQSk51bCt3b3c0bTZEc3F4Ym5pbmhzV0hsd2ZwMEplY3dRellwT0xtQ1E9PC9
kczpEaWdlc3RWYWx1ZT48L2RzOlJlZmVyZW5jZT48ZHM6UmVmZXJlbmNlIFR5cGU9Imh0dHA6Ly91cmkuZXRza
S5vcmcvMDE5MDMjU2lnbmVkUHJvcGVydGllcyIgVVJJPSIjeGFkZXMtaWQtZWExMGEwNTE3Y2JjN2Y1NDllYTN
lNjg1ODY3YWM5NWUiPjxkczpUcmFuc2Zvcm1zPjxkczpUcmFuc2Zvcm0gQWxnb3JpdGhtPSJodHRwOi8vd3d3L
nczLm9yZy9UUi8yMDAxL1JFQy14bWwtYzE0bi0yMDAxMDMxNSIvPjwvZHM6VHJhbnNmb3Jtcz48ZHM6RGlnZXN
0TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhMjU2Ii8+PGRzO
kRpZ2VzdFZhbHVlPnpUZlc5bjFUOWwwTHg2TlhGNUFUM1Btb3FLOWFHbUpIZlBDaXl0Z1JNeVU9PC9kczpEaWd
lc3RWYWx1ZT48L2RzOlJlZmVyZW5jZT48L2RzOlNpZ25lZEluZm8+PGRzOlNpZ25hdHVyZVZhbHVlIElkPSJ2Y
Wx1ZS1pZC1lYTEwYTA1MTdjYmM3ZjU0OWVhM2U2ODU4NjdhYzk1ZSI+Qy9FZnJvWmdGTkJvak40ZnpJd2UzVTR
ibDQ5S2xBbmhKSmI3elc0T2MxNWsweWpPSkZDcm9jY2JGV0U1eU52R3cyVHpxYVo0SVFYRjBKRllGM2IrNW5sa
G1EcTJacHBJYnNlOWY5M005cGU4cTVHaFBjWkRDV1FmNnp2TnNvUHRPYktzL04vWjlzODVVcmY3UGd1SWtVZlI
3eUJUaW5waGhJNVpuRHZuSnZsQ1RNRE5tYm4yMlBYSlYyc1IxWFpsa1BLWUsvVGhyOFdnSlcrUU9VREdTeGVST
mgyUWJPTllhR1FsUVJ6Vkhqb0c3emppVHM3UkdrR1ZVNGh3Q0pieW9TZThkd20zbkUxenVuQmp2TkRWenVqZVF
yZkhTSjNrQUsxbS9odkt4TnhlNXR4bkNmd0ZkNldDanpDWXBBaE9sSmI3MDJvV3RYam56azBHbkFZQndnPT08L
2RzOlNpZ25hdHVyZVZhbHVlPjxkczpLZXlJbmZvPjxkczpYNTA5RGF0YT48ZHM6WDUwOUNlcnRpZmljYXRlPk1
JSUQxRENDQXJ5Z0F3SUJBZ0lCQ2pBTkJna3Foa2lHOXcwQkFRc0ZBREJOTVJBd0RnWURWUVFEREFkbmIyOWtMV
05oTVJrd0Z3WURWUVFLREJCT2IzZHBibUVnVTI5c2RYUnBiMjV6TVJFd0R3WURWUVFMREFoUVMwa3RWRVZUVkR
FTE1Ba0dBMVVFQmhNQ1RGVXdIaGNOTVRjeE1ERTVNRGN5TXpFd1doY05NVGt3T0RFNU1EY3lNekV3V2pCUE1SS
XdFQVlEVlFRRERBbG5iMjlrTFhWelpYSXhHVEFYQmdOVkJBb01FRTV2ZDJsdVlTQlRiMngxZEdsdmJuTXhFVEF
QQmdOVkJBc01DRkJMU1MxVVJWTlVNUXN3Q1FZRFZRUUdFd0pNVlRDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBR
GdnRVBBRENDQVFvQ2dnRUJBSi9SVHV5WVRvR0RpbUZGR21STDhsN3lyRDZ1WDh1bkYzZkFmMGVTclpSRGZpS1R
3RjlPT3RVclc1OU5EMnNyQyt5aXBWSllHRlNBcTJxS0NuRlBMcXZOV0ZTbCtnZnJtWnNuN2tjUUJWdlFqQjljS
1pHMmc0VW55VUxCa3JQMFIvcW1pTmRuM0kzNHE2a0lBV3hXUnprUC9CaFAxdWVVNjBnUnhlV01HUGEyeVZabnh
KbDFUOEplSGkvSmpoN2tQSTgwR3V4UXJDQkg0eGJRWngvU1FpV2pJdDdwcm1WZ0crR0hoNHFpb0JGWWp2OGdQM
```

2ZMdExrM3ZGRHptRWFPeHRMMWRHQXdwcG83R0x1VXRVNkp4QXYwZm52eTZVT055QW4rK3V0OHFrOFU4WlM0Sk9
nRnpKdlZLT2N0UXBUVGptbitMbG9uMWlncmtiSEptN2lxYWJrQ0F3RUFBYU9CdkRDQnVUQU9CZ05WSFE4QkFmO
EVCQU1DQmtBd2dZY0dDQ3NHQVFVRkJ3RUJCSHN3ZVRBNUJnZ3JCZ0VGQlFjd0FZWXRhSFIwY0RvdkwyUnpjeTV
1YjNkcGJtRXViSFV2Y0d0cExxWmhZM1J2Y25rdmIyTnpjQzluYjI5a0xXTmhNRHdHQ0NzR0FRVUZCekFDaGpCb
2RIUndPaTh2WkhOekxtNXZkMmx1WVM1c2RTOXdhMmt0Wm1GamRHOXllUzlqY25RdloyOXZaQzFqWVM1amNuUXd
IUVlEVlIwT0JCWUVGTy9QL2lxRW92UW4zYXByN1VTT0tTUzdSTUU5TUEwR0NTcUdTSWIzRFFFQkN3VUFBNElCQ
VFBRTJMSjdKa1RlQ2lRT1plSTBTalcySnAwTFA2S1lpNWNOYzR6SW4wTlkwQ05UUkZzejdrRlc2QW9FVGhoSjk
zMmd5NmxSK0ZMR3BwS1NRNVZtUDBLd2JZV3g2MGFUSFJTbmtramRvZnlrYStoNitSbk1mRnl3NG9pZGVxdTBFW
HBMNFhtVFNQN3hPNi9PN2EzZk9kM01DUy9Udm4wQllmNVlTOFJuZXd4MHFBZk5hb3czYUhDMTEyQkFTMUFNZVV
Tc2x5QVBYMUNGZ2dtK25aUEgwcnVUL0NQVko5WlR6VFcyMlhLa1liaytHTVFEOGxRR1RwYTBzVnU1K2Z3Zm1JZ
28xZ1NqY20raXhKN04raDVtVXFZcE1Ydkp1TnJLUWwvSjA3RURWWmlrRWVnL2NQTkV2TE1XOXU5ckxqdU1rZWp
hQytETFUxRkpCZEpvd3FJS2NBakE8L2RzOlg1MDlDZXJ0aWZpY2F0ZT48ZHM6WDUwOUNlcnRpZmljYXRlPk1JS
UQ2akNDQXRLZ0F3SUJBZ0lCQkRBTkJna3Foa2lHOXcwQkFRc0ZBREJOTVJBd0RnWURWUVFEREFkeWIyOTBMV05
oTVJrd0Z3WURWUVFLREJCT2IzZHBibUVnVTI5c2RYUnBiMjV6TVJFd0R3WURWUVFMREFoUVMwa3RWRVZUVkRFT
E1Ba0dBMVVFQmhNQ1RGVXdIaGNOTVRjeE1ERTVNRGN5TWpVeFdoY05NVGt3T0RFNU1EY3lNalV4V2pCTk1SQXd
EZ1lEVlFRRERBZG5iMjlrTFdOaE1Sa3dGd1lEVlFRS0RCQk9iM2RwYm1FZ1UyOXNkWFJwYjI1ek1SRXdEd1lEV
lFRTERBaFFTMGt0VkVWVFZERUxNQWtHQTFVRUJoTUNURlV3Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR
3QXdnZ0VLQW9JQkFRQ2U4bjJoTDJrKzRRcklXUDJ6UmxMQkhBK1RGRDVtZlFrNWlna3plRzI4UDZSSXAxZlQwM
HdDQzk3MVRndktlZ0xyTmx5REducEFzQ2k1UDZndXd3dDk3NFhKSGJoTitZc0xJa2g3djRYbVVQSFpDcEpLS1h
ScCs1bThpS002cGJGSS8rOE9KQ0JYaDMxY3pHTFlnRUFnQ0ZkVTg5WXY5YTl2Z1FJVkQ3bko3aUFRV0xoSHJ6S
1lwSkQ0OEt2WklHMVJDNDhZNjhtNjFDZEdzenRVTHVHV1I1OGo5Zm5qanVRSTRITWNmY1ljK1pWRWR1dUp0bWp
1M3h4UkE1aGhIYkczaHN1NHpjSVJLd1pBT0hGcGJNVnZWVDVSZk9GTE9rNkt6WlR0NzFUSzVMbk5WN1lvSHc3O
XJXU29yRkxrRzRMVUxTR2d5bHllTVVUdHd5R25GeVpuQWdNQkFBR2pnZFF3Z2RFd0RnWURWUjBQQVFIL0JBUUR
BZ2VBTUVFR0ExVWRId1E2TURnd05xQTBvREtHTUdoMGRIQTZMeTlrYzNNdWJtOTNhVzVoTG14MUwzQnJhUzFtW
VdOMGIzSjVMMk55YkM5eWIyOTBMV05oTG1OeWJEQk1CZ2dyQmdFRkJRY0JBUVJBTUQ0d1BBWUlLd1lCQlFVSE1
BS0dNR2gwZEhBNkx5OWtjM011Ym05M2FXNWhMbXgxTDNCcmFTMW1ZV04wYjNKNUwyTnlkQzl5YjI5MExXTmhMb
U55ZERBZEJnTlZIUTRFRmdRVUhGUXMweWRjUDFSUHlvWXJ2bExHUjFaYksxZ3dEd1lEVlIwVEFRSC9CQVV3QXd
FQi96QU5CZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFIM0hkZkpQYkhPQ3BjRXBteHZaRi9VMjcreTB3VFd6aUo0a
3Z1Rnp5YmNMcjJyRWt3UkpldDBPaEZBMlBTSXFZZXc5S1lpb3BEd0VsOGQxSXA4L3k5Tk1kYU9VWUVpK2RTZzk
wMWNnVnhxRlFFRHJadUpWdEljQnh3MzBiNWFPMUE1V0FRRzhCMVhaNjI1K0NielRNQllOK0xoRHFZRWJhK1FXW
mdBR3BzWDFOS281TmxtK0wySmlVdng5QjlXcU95YkxZSWxWbmxuSGk3bFRJNDBjMjNTM2hTYVp6Z3lBdUFWR2N
TKzZFSldSc0dYNXJtaUE1MUNlTUhoMEtCdXRlL0FkczVOb0RteW93bHlhYU5vZHBTc2NiVWxIK0hneGlMVWRYN
0tJNDlabWRGSWtzUDB2Q1VvWFliWFlTekdmYmt2VGZ5SjQ5NXJzcktkaTcreWg0NlE9PTwvZHM6WDUwOUNlcnR
pZmljYXRlPjxkczpYNTA5Q2VydGlmaWNhdGU+TUlJRFZ6Q0NBaitnQXdJQkFnSUJBVEFOQmdrcWhraUc5dzBCQ
VEwRkFEQk5NUkF3RGdZRFZRUUREQWR5YjI5MExXTmhNUmt3RndZRFZRUUtEQkJPYjNkcGJtRWdVMjlzZFhScGI
yNXpNUkV3RHdZRFZRUUxEQWhRUzBrdFZFVlRWREVMTUFrR0ExVUVCaE1DVEZVd0hoY05NVGN3T1RFNU1EY3lNa
lF4V2hjTk1Ua3dPVEU1TURjeU1qUXhXakJOTVJBd0RnWURWUVFEREFkeWIyOTBMV05oTVJrd0Z3WURWUVFLREJ
CT2IzZHBibUVnVTI5c2RYUnBiMjV6TVJFd0R3WURWUVFMREFoUVMwa3RWRVZUVkRFTE1Ba0dBMVVFQmhNQ1RGV
XdnZ0VpTUEwR0NTcUdTSWIzRFFFQkFRVUFBNElCRHdBd2dnRUtBb0lCQVFEUmc0SDRvbEhveDFlNzljVUprSTV
1SitETGtYVnl2ZkExNWlWZGVJY3ZhS0xrUmdoYWlheTRsbmRJWTVGRjR0TVkwRWI2aW45Z1B2VzlnZytPMy9BM
HFUcHc0OXA5Z0FSdXE0SzJmNGFUZC8zUmdVem8wNHRXblJkbUg3Tm5Nc3ZKcmhHcGRvclpnejd5SmlHUVVWRjQ
4bFkzT0VLd3dCWUQzOGJER01UZG9jdGdrY2F6bThFVGF6M0hwQm9yRi9GM09nZ3JPNUc0SldtNGFuTlBvYUdZM
WZaR3ZJQ0RTNCtlejNlaElkNytobS80Sjkyc2hwUkRuMjl4djdra3g5VVBCQWVSYjZ3YzVhTkxmdGx2aEF4S0U
2bk5Dbk0wYXBvQmRCRGVuY3IzZk9SWlU0cmxxd1NsNkg2T3pseHFDdW9QQkp5R2tra3hvRzRabHVmWGhQV2JBZ
01CQUFHalFqQkFNQTRHQTFVZER3RUIvd1FFQXdJQkJqQWRCZ05WSFE0RUZnUVVZVVB6YmV6NXNiY0pIcys5OE1
HU2haEw3UEF3RHdZRFZSMFRBUUgvQkFVd0F3RUIvekFOQmdrcWhraUc5dzBCQVEwRkFBT0NBUUVBVG1QYVpTb
0dMNFg5UTFmOXhOa0NCYjZUQjlTUmEwZVVCKy9wUUVReXR5Rys5c3lFRkY4aGVmVjB6bGdGOUZqM1VwbWwyM0h
1dnZRQXk1YmE4dGxxWStMdE52THBRb1pHcXZEUDN0NkFlRDNONTQwNFNzd2FpT2tPL1gySmVZZzl3RDN4RU9na
kNSTVdyTU1FSWhxb1pOZXFIN2dLS1pKL3RHT01vSExjSHFXYVZmbGpqVmNUNnA0enI4bzB0MXl5T3AzNlNqVS9
LOHBNdFg0YU1PU05uUlpTdnl2a3F5Ly9pNHlFbmFRNnMvVks1eUgzYStXcENiTnpLQ0xmbTEzMS8rVUdZV1FOV
GIzWURYUUtGWmkwcnZoOGtodFFDeVEzYXVzQUZMdWsyc0FmSUszVGtIZml1ZnFZSXZsbEMzclZZQUo2TU9lWWl

```
XWGpTd1RrK25pVWFBPT08L2RzOlg1MDlDZXJ0aWZpY2F0ZT48L2RzOlg1MDlEYXRhPjwvZHM6S2V5SW5mbz48Z
HM6T2JqZWN0Pjx4YWRlczpRdWFsaWZ5aW5nUHJvcGVydGllcyB4bWxuczp4YWRlcz0iaHR0cDovL3VyaS5ldHN
pLm9yZy8wMTkwMy92MS4zLjIjIiBUYXJnZXQ9IiNpZC1lYTEwYTA1MTdjYmM3ZjU0OWVhM2U2ODU4NjdhYzk1Z
SI+PHhhZGVzOlNpZ25lZFByb3BlcnRpZXMgSWQ9InhhZGVzLWlkLWVhMTBhMDUxN2NiYzdmNTQ5ZWEzZTY4NTg
2N2FjOTVlIj48eGFkZXM6U2lnbmVkU2lnbmF0dXJlUHJvcGVydGllcz48eGFkZXM6U2lnbmluZ1RpbWU+MjAxO
C0wOS0yN1QxMTo1ODo0M1o8L3hhZGVzOlNpZ25pbmdUaW1lPjx4YWRlczpTaWduaW5nQ2VydGlmaWNhdGVWMj4
8eGFkZXM6Q2VydD48eGFkZXM6Q2VydERpZ2VzdD48ZHM6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL
3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI3NoYTEiLz48ZHM6RGlnZXN0VmFsdWU+aE5yb1k4cjFDQjU5Nmp
HQlBnUmdaRnZSRGJjPTwvZHM6RGlnZXN0VmFsdWU+PC94YWRlczpDZXJ0RGlnZXN0Pjx4YWRlczpJc3N1ZXJTZ
XJpYWxWMj5NRll3VWFSUE1FMHhFREFPQmdOVkJBTU1CMmR2YjJRdFkyRXhHVEFYQmdOVkJBb01FRTV2ZDJsdVl
TQlRiMngxZEdsdmJuTXhFVEFQQmdOVkJBc01DRkJMU1MxVVJWTlVNUXN3Q1FZRFZRUUdFd0pNVlFJQkNnPT08L
3hhZGVzOklzc3VlclNlcmlhbFYyPjwveGFkZXM6Q2VydD48L3hhZGVzOlNpZ25pbmdDZXJ0aWZpY2F0ZVYyPjw
veGFkZXM6U2lnbmVkU2lnbmF0dXJlUHJvcGVydGllcz48eGFkZXM6U2lnbmVkRGF0YU9iamVjdFByb3BlcnRpZ
XM+PHhhZGVzOkRhdGFPYmplY3RGb3JtYXQgT2JqZWN0UmVmZXJlbmNlPSIjci1pZC0xIj48eGFkZXM6TWltZVR
5cGU+dGV4dC9wbGFpbjwveGFkZXM6TWltZVR5cGU+PC94YWRlczpEYXRhT2JqZWN0Rm9ybWF0PjwveGFkZXM6U
2lnbmVkRGF0YU9iamVjdFByb3BlcnRpZXM+PC94YWRlczpTaWduZWRQcm9wZXJ0aWVzPjwveGFkZXM6UXVhbGl
meWluZ1Byb3BlcnRpZXM+PC9kczpPYmplY3Q+PGRzOk9iamVjdCBJZD0iby1pZC0xIj5hR1ZzYkc4PTwvZHM6T
2JqZWN0PjwvZHM6U2lnbmF0dXJlPg==",
    "digestAlgorithm" : null,
    "name" : "hello-signed-xades.xml"
  },
  "originalDocuments" : null,
  "policy" : null,
  "signatureId" : "id-ea10a0517cbc7f549ea3e685867ac95e"
}
```


Response

```
HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 06:08:46 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
Transfer-Encoding: chunked
Content-Length: 3

[ ]
```

REST certificate validation service

Validate a certificate

This service allows a certificate validation (provided in a binary format).

Request

```
POST /services/rest/certificate-validation/validateCertificate HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 2156

{
  "certificate" : {
    "encodedCertificate" :
"MIIC6jCCAdKgAwIBAgIGLtYU17tXMA0GCSqGSIb3DQEBCwUAMDAxGzAZBgNVBAMMElJvb3RTZWxmU2lnbmVkR
mFrZTERMA8GA1UECgwIRFNTLXRlc3QwHhcNMTcwNjA4MTEyNjAxWhcNNDcwNzA0MDc1NzI0WjAoMRMwEQYDVQQ
DDApTaWduZXJGYWtlMREwDwYDVQQKDAhEU1MtdGVzdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
MI3kZhtnipn+iiZHZ9ax8FlfE5Ow/cFwBTfAEb3R1ZQUp6/BQnBt7Oo0JWBtc9qkv7JUDdcBJXPV5QWS5AyMPH
pqQ75Hitjsq/Fzu8eHtkKpFizcxGa9BZdkQjh4rSrtO1Kjs0Rd5DQtWSgkeVCCN09kN0ZsZ0ENY+Ip8QxSmyzt
sStkYXdULqpwz4JEXW9vz64eTbde4vQJ6pjHGarJf1gQNEc2XzhmI/prXLysWNqC7lZg7PUZUTrdegABTUzYCR
J1kWBRPm4qo0LN405c94QQd45a5kTgowHzEgLnAQI28x0M3A59TKC+ieNc6VF1PsTLpUw7PNI2VstX5jAuasCA
wEAAaMSMBAwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBCwUAA4IBAQCK6LGA01TR+rmU8p6yhAi4OkDN2b1
dbIL8l8iCMYopLCxx8xqq3ubZCOxqh1X2j6pgWzarb0b/MUix00IoUvNbFOxAW7PBZIKDLnm6LsckRxs1U32sC
9d1LOHe3WKBNB6GZALT1ewjh7hSbWjftlmcovq+6eVGA5cvf2u/2+TkKkyHV/NR394nXrdsdpvygwypEtXjetz
D7UT93Nuw3xcV8VIftIvHf9LjU7h+UjGmKXG9c15eYr3SzUmv6kyOI0Bvw14PWtsWGl0QdOSRvIBBrP4adCnGT
gjgjk9LTcO8B8FKrr+8lHGuc0bp4lIUToiUkGILXsiEeEg9WAqm+XqO"
  },
  "certificateChain" : [ {
    "encodedCertificate" :
"MIIC6jCCAdKgAwIBAgIGLtYU17tXMA0GCSqGSIb3DQEBCwUAMDAxGzAZBgNVBAMMElJvb3RTZWxmU2lnbmVkR
mFrZTERMA8GA1UECgwIRFNTLXRlc3QwHhcNMTcwNjA4MTEyNjAxWhcNNDcwNzA0MDc1NzI0WjAoMRMwEQYDVQQ
DDApTaWduZXJGYWtlMREwDwYDVQQKDAhEU1MtdGVzdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
MI3kZhtnipn+iiZHZ9ax8FlfE5Ow/cFwBTfAEb3R1ZQUp6/BQnBt7Oo0JWBtc9qkv7JUDdcBJXPV5QWS5AyMPH
pqQ75Hitjsq/Fzu8eHtkKpFizcxGa9BZdkQjh4rSrtO1Kjs0Rd5DQtWSgkeVCCN09kN0ZsZ0ENY+Ip8QxSmyzt
sStkYXdULqpwz4JEXW9vz64eTbde4vQJ6pjHGarJf1gQNEc2XzhmI/prXLysWNqC7lZg7PUZUTrdegABTUzYCR
J1kWBRPm4qo0LN405c94QQd45a5kTgowHzEgLnAQI28x0M3A59TKC+ieNc6VF1PsTLpUw7PNI2VstX5jAuasCA
wEAAaMSMBAwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBCwUAA4IBAQCK6LGA01TR+rmU8p6yhAi4OkDN2b1
dbIL8l8iCMYopLCxx8xqq3ubZCOxqh1X2j6pgWzarb0b/MUix00IoUvNbFOxAW7PBZIKDLnm6LsckRxs1U32sC
9d1LOHe3WKBNB6GZALT1ewjh7hSbWjftlmcovq+6eVGA5cvf2u/2+TkKkyHV/NR394nXrdsdpvygwypEtXjetz
D7UT93Nuw3xcV8VIftIvHf9LjU7h+UjGmKXG9c15eYr3SzUmv6kyOI0Bvw14PWtsWGl0QdOSRvIBBrP4adCnGT
gjgjk9LTcO8B8FKrr+8lHGuc0bp4lIUToiUkGILXsiEeEg9WAqm+XqO"
  } ],
  "validationTime" : null
}
```

Response

```
HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 06:08:46 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
```

```
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
Transfer-Encoding: chunked
Content-Length: 6315

{
  "diagnosticData" : {
    "DocumentName" : null,
    "ValidationDate" : "2019-08-23T06:08:46",
    "ContainerInfo" : null,
    "Signature" : null,
    "Certificate" : [ {
      "Id" : "C-02F3EBCA0163274253BC809D27498DD41BB0316D7E6B066960115DE155589D9C",
      "SubjectDistinguishedName" : [ {
        "value" : "o=dss-test,cn=signerfake",
        "Format" : "CANONICAL"
      }, {
        "value" : "O=DSS-test,CN=SignerFake",
        "Format" : "RFC2253"
      } ],
      "IssuerDistinguishedName" : [ {
        "value" : "o=dss-test,cn=rootselfsignedfake",
        "Format" : "CANONICAL"
      }, {
        "value" : "O=DSS-test,CN=RootSelfSignedFake",
        "Format" : "RFC2253"
      } ],
      "SerialNumber" : 51497007561559,
      "CommonName" : "SignerFake",
      "Locality" : null,
      "State" : null,
      "CountryName" : null,
      "OrganizationName" : "DSS-test",
      "GivenName" : null,
      "OrganizationalUnit" : null,
      "Surname" : null,
      "Pseudonym" : null,
      "Email" : null,
      "aiaUrl" : [ ],
      "crlUrl" : [ ],
      "ocspServerUrl" : [ ],
      "Source" : [ "OTHER" ],
      "NotAfter" : "2047-07-04T07:57:24",
      "NotBefore" : "2017-06-08T11:26:01",
      "PublicKeySize" : 2048,
      "PublicKeyEncryptionAlgo" : "RSA",
      "KeyUsage" : [ "keyCertSign", "crlSign" ],
      "extendedKeyUsagesOid" : [ ],
      "IdPkixOcspNoCheck" : false,
```

```
      "BasicSignature" : {
        "EncryptionAlgoUsedToSignThisToken" : "RSA",
        "KeyLengthUsedToSignThisToken" : "?",
        "DigestAlgoUsedToSignThisToken" : "SHA256",
        "MaskGenerationFunctionUsedToSignThisToken" : null,
        "SignatureIntact" : false,
        "SignatureValid" : false
      },
      "SigningCertificate" : null,
      "ChainItem" : [ ],
      "Trusted" : false,
      "SelfSigned" : false,
      "certificatePolicy" : [ ],
      "qcStatementOid" : [ ],
      "qcTypeOid" : [ ],
      "TrustedServiceProvider" : [ ],
      "CertificateRevocation" : [ ],
      "Base64Encoded" : null,
      "DigestAlgoAndValue" : {
        "DigestMethod" : "SHA256",
        "DigestValue" : "AvPrygFjJ0JTvICdJ0mN1BuwMW1+awZpYBFd4VVYnZw="
      }
    } ],
    "Revocation" : [ ],
    "Timestamp" : null,
    "OrphanToken" : null,
    "SignerData" : null,
    "TrustedList" : [ ],
    "ListOfTrustedLists" : null
  },
  "simpleCertificateReport" : {
    "ChainItem" : [ {
      "id" : "C-02F3EBCA0163274253BC809D27498DD41BB0316D7E6B066960115DE155589D9C",
      "subject" : {
        "commonName" : "SignerFake",
        "surname" : null,
        "givenName" : null,
        "pseudonym" : null,
        "organizationName" : "DSS-test",
        "organizationUnit" : null,
        "email" : null,
        "locality" : null,
        "state" : null,
        "country" : null
      },
      "issuerId" : null,
      "notBefore" : "2017-06-08T11:26:01",
      "notAfter" : "2047-07-04T07:57:24",
      "keyUsage" : [ "keyCertSign", "crlSign" ],
      "extendedKeyUsage" : null,
      "ocspUrl" : null,
```

```
      "crlUrl" : null,
      "aiaUrl" : null,
      "cpsUrl" : null,
      "pdsUrl" : null,
      "qualificationAtIssuance" : "N/A",
      "qualificationAtValidation" : "N/A",
      "revocation" : {
        "productionDate" : null,
        "revocationDate" : null,
        "revocationReason" : null
      },
      "trustAnchor" : null,
      "Indication" : "INDETERMINATE",
      "SubIndication" : "NO_CERTIFICATE_CHAIN_FOUND"
    } ],
    "ValidationTime" : "2019-08-23T06:08:46"
  },
  "detailedReport" : {
    "Signatures" : null,
    "Certificate" : {
      "ValidationCertificateQualification" : [ ],
      "Constraint" : [ {
        "Name" : {
          "value" : "Is the result of the Basic Building Block acceptable?",
          "NameId" : "BBB_ACCEPT"
        },
        "Status" : "WARNING",
        "Error" : null,
        "Warning" : {
          "value" : "The result of the Basic Building Block is not acceptable!",
          "NameId" : "BBB_ACCEPT_ANS"
        },
        "Info" : null,
        "AdditionalInfo" : null,
        "Id" : null
      } ],
      "Conclusion" : {
        "Indication" : "INDETERMINATE",
        "SubIndication" : null,
        "Errors" : [ ],
        "Warnings" : [ {
          "value" : "The result of the Basic Building Block is not acceptable!",
          "NameId" : "BBB_ACCEPT_ANS"
        } ],
        "Infos" : null
      },
      "Title" : "Certificate Qualification"
    },
    "BasicBuildingBlocks" : [ {
      "FC" : null,
      "ISC" : null,
```

```
      "VCI" : null,
      "XCV" : {
        "SubXCV" : [ ],
        "Constraint" : [ {
          "Name" : {
            "value" : "Can the certificate chain be built till the trust anchor?",
            "NameId" : "BBB_XCV_CCCBB"
          },
          "Status" : "NOT OK",
          "Error" : {
            "value" : "The certificate chain is not trusted, there is no trusted
anchor.",
            "NameId" : "BBB_XCV_CCCBB_ANS"
          },
          "Warning" : null,
          "Info" : null,
          "AdditionalInfo" : null,
          "Id" : null
        } ],
        "Conclusion" : {
          "Indication" : "INDETERMINATE",
          "SubIndication" : "NO_CERTIFICATE_CHAIN_FOUND",
          "Errors" : [ {
            "value" : "The certificate chain is not trusted, there is no trusted
anchor.",
            "NameId" : "BBB_XCV_CCCBB_ANS"
          } ],
          "Warnings" : null,
          "Infos" : null
        },
        "Title" : "X509 Certificate Validation"
      },
      "CV" : null,
      "SAV" : null,
      "PSV" : null,
      "PCV" : null,
      "VTS" : null,
      "CertificateChain" : null,
      "Conclusion" : {
        "Indication" : "INDETERMINATE",
        "SubIndication" : "NO_CERTIFICATE_CHAIN_FOUND",
        "Errors" : [ {
          "value" : "The certificate chain is not trusted, there is no trusted
anchor.",
          "NameId" : "BBB_XCV_CCCBB_ANS"
        } ],
        "Warnings" : null,
        "Infos" : null
      },
      "Id" : "C-02F3EBCA0163274253BC809D27498DD41BB0316D7E6B066960115DE155589D9C",
      "Type" : "CERTIFICATE"
```

```
    } ],
    "TLAnalysis" : [ ]
  }
}
```

REST remote timestamp service

Get Timestamp Response

This service allows a remote timestamp creation. The method takes as an input the digest to be timestamped and digest algorithm that has been used for the digest value computation. The output of the method is the generated timestamp's binaries.

Request

```
POST /services/rest/timestamp-service/getTimestampResponse HTTP/1.1
Accept: application/json, application/javascript, text/javascript, text/json
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 73

{
  "algorithm" : "SHA1",
  "value" : "Ir3ZcUPfazmmWvrSr2vYvyDNT3s="
}
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 10 Sep 2019 08:28:36 GMT
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Server: ESIG-DSS
Content-Type: application/json
Transfer-Encoding: chunked
Content-Length: 443

{
  "binaries":
"MIIGPAYJKoZIhvcNAQcCoIIGLTCCBikCAQMxCzAJBgUrDgMCGgUAMGIGCyqGSIb3DQEJEAEEoFMEUTBPAgEBB
gMqAwQwITAJBgUrDgMCGgUABBQivdlxQ99rOaZa+tKva9i/IM1PewIRAJUnHe2aMlCOWu76E5KQRIkYDzIwMTk
wOTEwMDgyODM2WqCCA3IwggNuMIICVqADAgECAgFkMA0GCSqGSIb3DQEBCwUAMFUxGDAWBgNVBAMMD3NlbGYtc
2lnbmVkLXRzYTEZMBcGA1UECgwQTm93aW5hIFNvbHV0aW9uczERMA8GA1UECwwIUEtJLVRFU1QxCzAJBgNVBAY
TAkxVMB4XDTE4MDkyMDA3MTMyNloXDTIwMDcyMDA3MTMyNlowVTEYMBYGA1UEAwwPc2VsZi1zaWduZWQtdHNhM
RkwFwYDVQQKDBBOb3dpbmEgU29sdXRpb25zMREwDwYDVQQLDAhQS0ktVEVTVDELMAkGA1UEBhMCTFUwggEiMA0
GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDAGRWF8H6gCPwzhLxd48v/lOf93ieRBRDxG/v59cvm1QcrTNcaX
5QtkMAzxRPu7GB9VS0Hr9C2IiYn+o8uA/GeEdeabKWiXFWSRu4+vOBt4gqo1OqCTjWkSn1XmfifI4hVW9kcWiY
63I7EKJJbgVcni1Kl6L2NAi9hdYIxGScKIAhqUcFWCGaA3kRH4v4WYixpTJ7syHT+iU0BXvrGZzbERZvK2R6bl
j2QRB/07Fo6z4YfTHsuGZ0HGG7UpR805LfgPjVReGvlQyfXqo+fbE5Hybiu9b8zBNpciqpXraWiyia5cOOYZr5
CSIRzDvxvds/R9Frl2QJCwR+Y3futryJVAgMBAAGjSTBHMA4GA1UdDwEB/wQEAwIHgDAWBgNVHSUBAf8EDDAKB
ggrBgEFBQcDCDAdBgNVHQ4EFgQUZ4HHYmM3oSJdBNSGGx9LmzTL85YwDQYJKoZIhvcNAQELBQADggEBAIxxc3H
Kee+APEuRPFbqlQfDitVYY/6JiJQjN34/mhfEt7XzpD03mhFOZc2itHWL/m6rDUCIk7wMPGUelZlqNNs9iuBqJ
0riWLCMckbJNO0UXU5af1LSRLBdWUlkfcusSylePuovl1OrNxwX1ZdSAzubyBzTJ89p4Y3BINxHHbEYkgbDIlb
12Ord//pTEmKH0PDxobOvOauCDHuhVSc5GBOhyzy9DTmhDxw2TTjg9cjuUOMD4YYHBsYAcM3T1duHqN2652cBf
FNouyEyEMuavVphkv1wS4S66v15yefe/nb+wpgdxvsMp3tWLzQivP6p2PWCbdqor/LwD+jhXEw5cwMxggI7MII
CNwIBATBaMFUxGDAWBgNVBAMMD3NlbGYtc2lnbmVkLXRzYTEZMBcGA1UECgwQTm93aW5hIFNvbHV0aW9uczERM
A8GA1UECwwIUEtJLVRFU1QxCzAJBgNVBAYTAkxVAgFkMAkGBSsOAwIaBQCggbcwGgYJKoZIhvcNAQkDMQ0GCyq
GSIb3DQEJEAEEMBwGCSqGSIb3DQEJBTEPFw0xOTA5MTAwODI4MzZaMCMGCSqGSIb3DQEJBDEWBBQ2cj+v31z4A
Znky8PoqEN/78VE5jApBgkqhkiG9w0BCTQxHDAaMAkGBSsOAwIaBQChDQYJKoZIhvcNAQEBBQAwKwYLKoZIhvc
NAQkQAgwxHDAaMBgwFgQUQbys1f8dQklOKjibmaM589F50/wwDQYJKoZIhvcNAQEBBQAEggEAh1be2gI7ovwDF
LPsjNYnVsjXCiVCUlZR+GJ5p+o1MQhB2j0b9qJtE4KxNFAu9Vm6Jk+ffLK2okcgPRyHGEHMekeLPIxUerBX7Au
GdZR1JJl95em87Pkb6gp8OuWuHPL9bLw0+daHF9n4giLVHZ0BZ67HNQOgNxaLvqHGVVH9adwRyk2or+884/bEv
qPINQPdEHc+pUqlBirzTaWcxJCEALvYoRsnkdO7/45eDk4TwfF0qb5r9UCxPmRaLbhfHltf78PpZ4BBV4oAl8G
Bwqt1VzSnCOWxrfj69ox6jWoLV8jlxP/14Mi79n9nyhNxTr/DodBqkFuWO/GCJJfQjT9PbA=="
}
```