Contents

1 Introduction

SiVa is digital signature validation web service that provides SOAP and JSON API to validate following file types:

- Older Estonian digital signature files with DDOC extension
- BDOC containers with `TimeMark` and `TimeStamp` signatures
- Digitally signed PDF files
- X-Road security server ASiCE signature containers

Architecture document main purpose is to give overview what SiVa is. Give an overview of it's internal processes and provide information when deploying it to production environment.

1.1 SiVa architecture document sections overview

Below list will give You an overview of what each section of the SiVa architecture document will cover:

- **Overview** - gives overview what SiVa is and it's main features.
- **Regulatory environment** - legal analysis and standards that are used when building SiVa application
- **Component diagram** - gives overview of main SiVa subsystems and and and base validation Java libraries used for different validation services

- **Deployment view** - gives general overview of servers required when deploying SiVa validation web service into production
- **Interfaces** - Description of SiVa SOAP and JSON API request and response
- **Use cases** - describes main processes in SiVa validation web service
- **Deploying** - how to build, deploy and configure SiVa web service
- **Logging** - how to configure and setup SiVa validation service logging support
- **QA Strategy** - overview of quality assurance strategy
- **Test Plan** - overview of test planning

1.2 Download documentation

- **Download SiVa documentation as PDF**

2 Version info

Version number	Change date	Author	Description
0.1	06.05.2016	Mihkel Selgal	Initial SiVa architecture document

3 Definitions

DSS Digital Signature Services is Java library to sign and validate European digital signature formats
BDOC Not defined
DDOC Not defined
PDF Portable document format is a file format that provides an electronic image of text or text and graphics that looks like a printed document and can be viewed, printed, and electronically transmitted.
SiVa is RESTful web service providing digital signature validation services for BDOC, DDOC, PDF and X-Road files
JVM The Java Virtual Machine is the runtime engine of the Java Platform, which allows any program written in Java or other language compiled into Java bytecode to run on any computer that has a native JVM.
JAR Java Archive is a package file format typically used to aggregate many Java class files and associated metadata and resources (text, images, etc.) into one file to distribute application software or libraries on the Java platform.
Fat JAR The fat JAR is the JAR, which contains classes from all the libraries, on which your project depends and, the classes of built project.
Spring Boot is a framework from the team at Pivotal, designed to simplify the bootstrapping and development of a new Spring application. The framework takes an opinionated approach to configuration, freeing developers from the need to define boilerplate configuration
Linux an operating system, based on UNIX, that runs on many different hardware platforms and whose source code is available to the public.
PüPKI Not defined
PostgreSQL is an open source relational database management system (DBMS) developed by a worldwide team of volunteers. PostgreSQL is not controlled by any corporation or other private entity and the source code is available free of charge.

4 SiVa Service Overview

SiVa (Signature Validation) web service is continued development of PDF Validation web service. Service provides a JSON and SOAP based API web interface which purpose is to validate signatures in digitally signed BDOC, DDOC, PDF and X-Road ASiCE files according to validation policy (described in the Validation Policy section).

SiVa uses following Java libraries and command line utilities:

- EU DSS (Digital Signature Service) library is chosen for digitally signed PDF file validation
- DigiDoc4J Java library to validate BDOC containers. Supported signature types are `TimeStamp` and `TimeMark`
- JDigiDoc Java library is used to validate DDOC files starting from version
- X-Road ASiCE containers are validated using X-Road security server project provided command line utility

4.1 Validation libraries

4.1.1 DigiDoc4j EU DSS fork

DigiDoc4J EU DSS fork library for PDF files was chosen because it all the main validation constrains already provided and all new constraints can be added easily. For more information on EU DSS, see: https://joinup.ec.europa.eu/asset/sd-dss/description.

SiVa will use the following functionality of EU DSS library:

- PaDES Validation Functionality
- TSL loading functionality

4.1.2 DigiDoc4J

DigiDoc4J will be used to validate both `TimeMark` and `TimeStamp` based BDOC containers. DigiDoc4J was chosen because it's only Java library that can validate Estonian BDOC files according to SiVa validation policy. For more information on DigiDoc4J: https://github.com/open-eid/digidoc4j

SiVa will use the following functionality of DigiDoc4J:

- BDOC validation functionality

4.1.3 JDigiDoc

JDigiDoc provides support for DDOC files the library was chosen because it provides most complete support for all required DDOC versions. Read more about JDigiDoc: https://github.com/open-eid/jdigidoc

SiVa will use the following functionality of JDigiDoc:

- DDOC validation functionality

4.1.4 X-Road signature validation utility

X-Road signature validation utility is command line tool to validate X-Road Security server generated ASiCe files. The utility was chosen because it's only available packaged to tool to validate X-Road signature files.

4.2 Main features of SiVa validation service:

- SiVa SOAP ETSI compliant API to validate all supported signatures.
- SiVa REST ETSI compliant API to validate all supported signatures.
- SiVa handles files in PDF-format version 1.7 and later, signed with PadES-profile signatures.
- Service handles DDOC files starting from version 1.0 or later
- Service supports BDOC files starting from version 2.1 or later
- Service supports X-Road 6 security server ASiCE containers
- SiVa uses European Commission's TSL (Trusted Service Status List) for certificate chain validation for PDF and BDOC files.
  - European Commission's TSL contains references to TSLs of European Union's member states and members of the European Economic Area. This allows the PDF Validator to validate signature that has been signed with certificates issued in any of European Union's member states.
  - During the validation process, a certificate chain is created from signer's certificate up to the trust anchor (national trust list referenced by the central European Commission's trust list) for all certificates included in the signature (i.e. the signer's certificate, OCSP service's certificate, time-stamping Service's certificate).
- SiVas for DDOC and X-Road signature containers will use configured list certificates.
- Signatures with PadES-LT and PadES-LTA profile are supported.
- BDOC signatures with type BDOC-TM and BDOC-TS are supported

At the time of creating the current documentation, it is expected that SiVa will be used by the following applications:

- DigiDoc3 Client application
- Third party document management applications

5 Regulatory environment

!!! note Regulatory environment section will be added when analysis will be completed

6 Component diagram

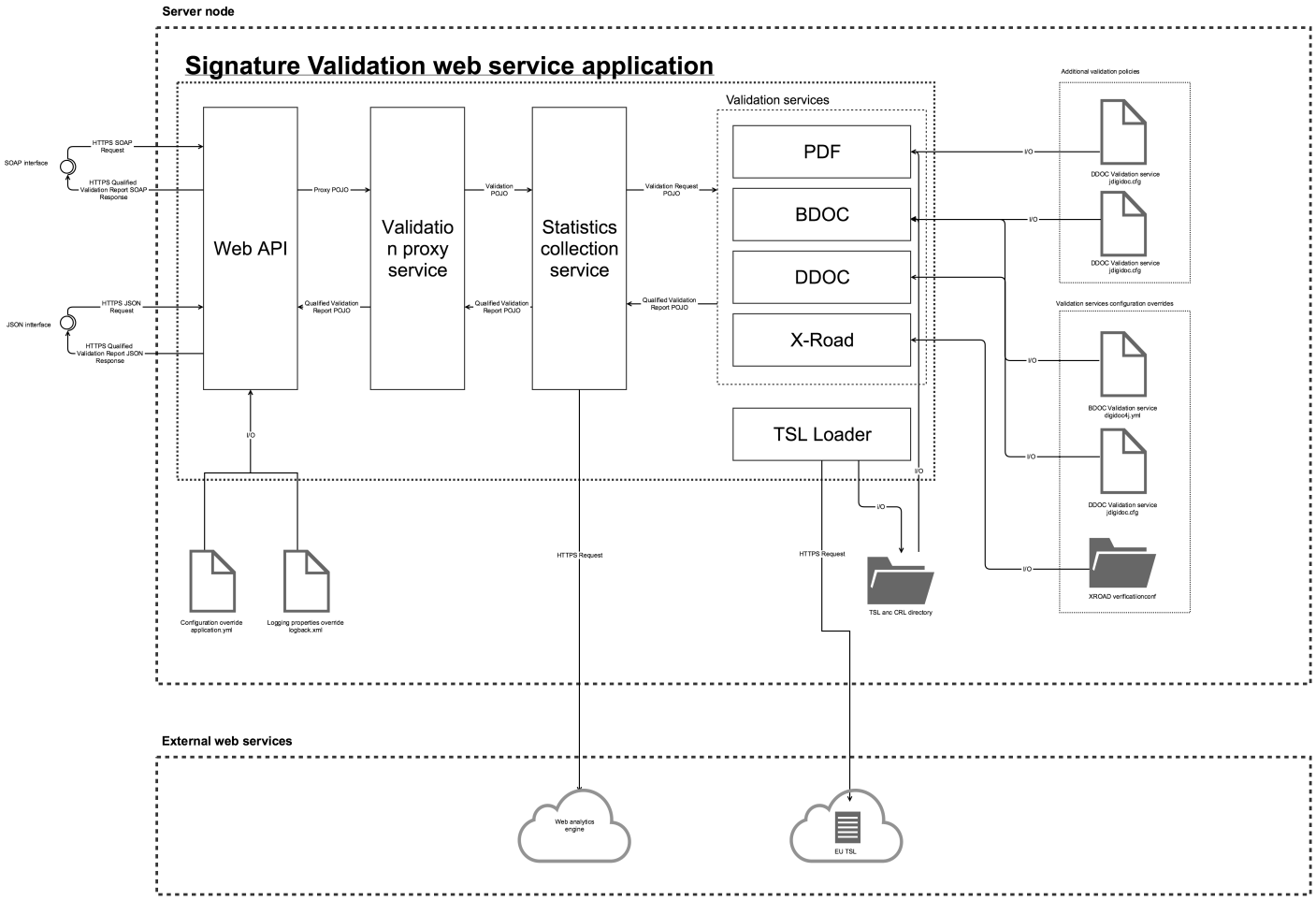

Figure 1: SiVa component diagram

6.1 Web API

Web API is standard Spring Boot Web application module inside SiVa webapp it will take in JSON or SOAP requests sent by systems that are integrated with SiVa web service API. The incoming requests will be converted to **SiVa Proxy Module** Java request objects. Web API module also does basic validation of incoming requests. Checking that all the required fields are present and document type is correct.

When validation has been completed by proxy selected validation service the returned qualified validation report Java object will be marshalled based on incoming request to JSON or SOAP and returned to client that requested the validation.

6.1.1 External configuration resources

- Optionally SiVa webapp can load in configuration file (i.e application.yml) at application startup time. Configuration file can control Spring Boot configuration options in addition to SiVa application specific options.
- Optionally SiVa webapp can also load in logging configuration options

6.2 Validation proxy service

Validation proxy service or validation service selector is Spring Boot module that will take the Web API sent request object and try to find matching validation service based on the `documentType`inside the request object. When matching validation service have been found the proxy request is converted to validation request and sent to matched validation service.

When no matching validation service has not been found exception is raised and error object is returned to Web API module. On successful validation the qualified validation report Java object sent from validation service is returned to Web API module.

!!! note Validation services can be added dynamically to SiVa by conforming to pattern `documentType + "ValidationService"` and new validation service module must be Maven dependency of `siva-validation-proxy`. Example would be `BDOCValidationService`.

6.3 Validation reporting service

Validation reporting service is optional module that can be turned on or off using configuration file. It's Spring Boot module and main purpose is to collect data about: incoming request, validation reports and errors that have been reported during validation process.

When HTTP authentication header have been set the reporting service will also collect its and adds to required statistics reports.

After the report object have been created the data will be sent to configured reporting service. SiVa is preconfigured to work with Google Analytics.

6.4 TSL Loader

TSL loader loads in contents of TSL file from given URL in online mode or from directory when using offline mode in predefined interval.

6.5 Validation services

All validation services use different Java library to validate given document in request. The used validation library is described in each of the validation service section.

Common process that all validation services do with proxy forwarded validation process is:

- Convert the Base64 encoded document into `InputStream` byte array
- Check that given document is correct format (i.e valid BDOC). If not then error is thrown and validation process is terminated.
- After validation of signatures has been completed the validation service starts to build qualified validation report
- Validation report is created even validation `FAILED` or ended with `INDETERMINATE` result

6.5.1 PDF Validation service

PDF or PaDES as known in DSS validation service uses Digidoc4J DSS fork Java library PaDES validation functionality using the validation policy that complies with Estonian laws and regulations.

Configurable functionality:

- Possibility to add additional validation policies using SiVa `application.yml` configuration section.

6.5.2 BDOC validation service

BDOC for ASiC compliant containers both TM and TS will latest Maven released **DigiDoc4J** library

6.5.3 DDOC Validation service

DDOC for previous generation digitally signed files will use latest Maven release of **JDigiDoc**

6.5.4 X-Road validation service

X-Road containers are similar to ASiCE containers but are **not** valid ASiCE containers. There we could not use DSS nor DigiDoc4J provided ASiCE validation functionality but need to X-Road developed `asicverifier` Java command line utility to validate these containers.

Source code for `asicverifier` can be found in GitHub xroad-public repository*[]: `Asicverfier` has been integrated into SiVa as Java library. Making possible to use all the Java libraries packaged into `asicverifier` fat JAR.

Configurable functionality:

- In SiVa configuration `application.yml` file You can define alternative location for `globalconf` directory to be loaded in using input stream

7 Deployment view

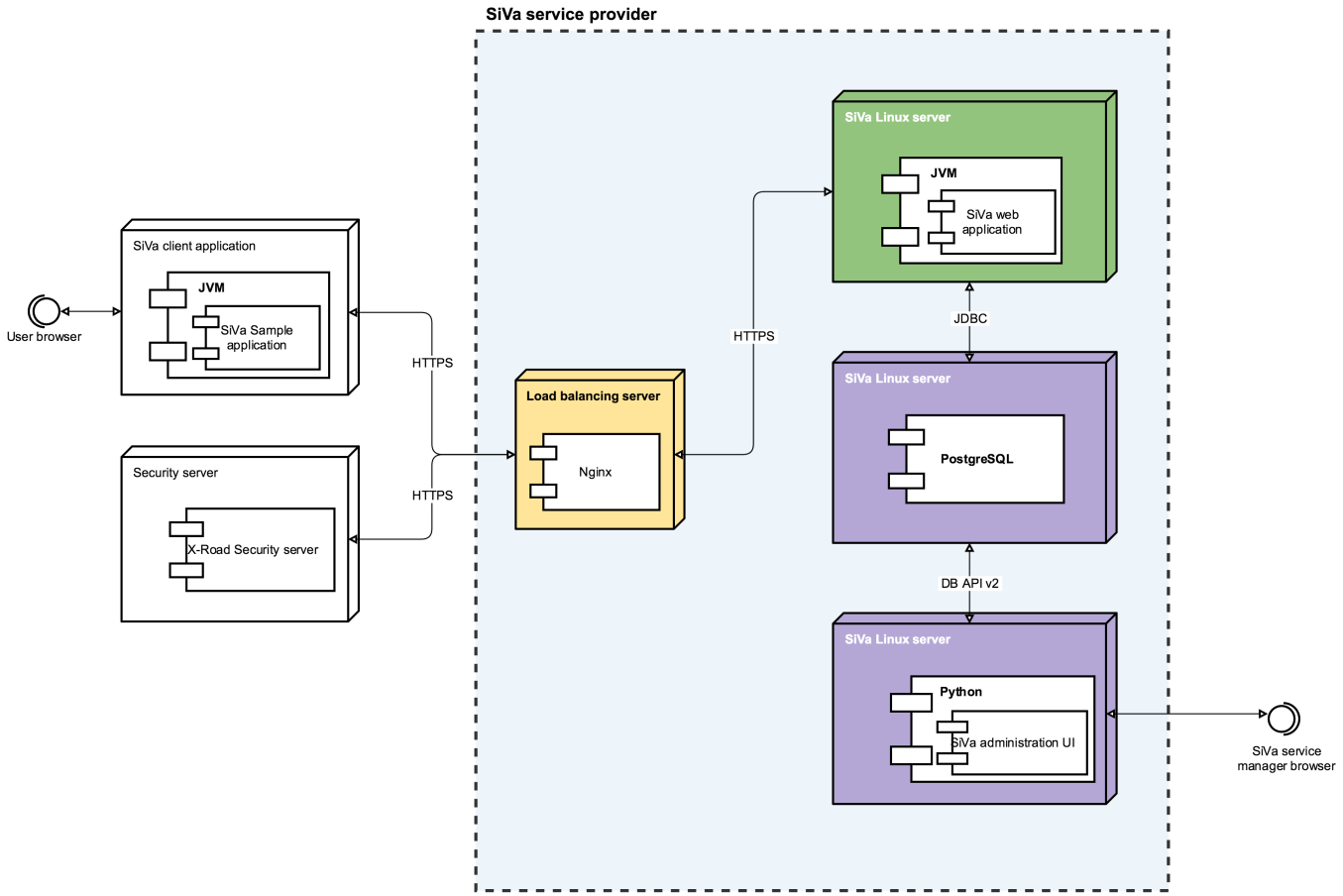

Figure 2: SiVa Deployment view

7.1 Load balancer

Load balancer can distribute traffic between SiVa nodes when there is more then one instance running. SiVa do not set any specific requirements for load balancer but in diagram the Nginx reverse proxy option is shown.

7.2 SiVa application server

SiVa validation service server that will provide the service needs to have JVM installed) both more commonly known options Oracle or OpenJDK are supported. SiVa application is built as executable JAR and can be configured like system service on Linux operating system.

Read more about running Spring Boot applications as Linux system service

SiVa validation service can run in cluster because it does not keep or create any sessions with client application or service.

!!! note The single executable JAR option may change in the future because we are considering isolating each validation service and SiVa web application into separate JVM instance

7.3 Database server

SiVa database server requirements are dictated by PüPKI application and from these requirements needs to be PostgreSQL. Currently there are no special requirements for database server nor database setup.

!!! note Database requirements section will be updated when analysis and development will begin on modification of PüPKI application

!!! development It may be possible that we will build administration user interface as part of SiVa project so the database and type of database RDBMS or NoSQL may change in the future

7.4 SiVa administration server

SiVa administration will use PüPKI administration user interface to manage authorized service users and collect information about basic service usage by authorized clients.

Only requirement currently known is that Python programming language support must be present in the server.

!!! note More detailed information about SiVa administration server setup will be provided when development and analysis will begin for administration service

!!! development There is alternative option that we will build administration user interface from scratch using Java and Spring Boot

8 Interfaces

9 Interface description

9.1 REST JSON validation service

9.1.1 REST JSON Endpoint

```
POST https://<server url>/validate
```

9.1.2 JSON Request

Parameter	Type	Mandatory	Description
`document`	String	•	Base64 encoded string of digitally signed file
`filename`	String	•	Filename of the digitally signed file (i.e `sample.bdoc`

Parameter	Type	Mandatory	Description
documentType	String	•	Validation service to use for validation
signaturePolicy	String	•	If You want to use alternative validation policy. Then You signaturePolicy for that.

9.1.2.1 Sample request

```
{
  "filename":"sample.ddoc",
  "documentType":"DDOC",
  "document":"PD94bWwgdmVyc2lvbj0iMS4....",
  "signaturePolicy": "EU"
}
```

9.1.3 JSON Response

Parameter	Type	Description
policy	Object	Version of SiVa validation policy
policy.policyVersion	String	Version of SiVa validation policy
policy.policyName	String	Name of policy used for validation
policy.policyDescription	String	Short description of validation used
policy.policyUrl	String	URL where the validation service's signature policy document can be downloaded. The validation policy document shall include information about validation of all the document formats, including the different validation policies that are used in case of different file formats and base libraries.
signaturesCount	Number	Number of signatures found inside digitally signed file
validSignaturesCount	Number	Signatures count that have validated to TOTAL-PASSED
validationTime	Date	Time of validating the signature by the service.
documentName	String	Digitally signed document filename
signatures	Array	Collection of signatures found digitally signed document
signatures[0]	Object	Signature information object
signatures[0].claimedSigningTime	Data	Time signature was given
signatures[0].errors	Array	Information about validation error(s). Array of error message, as returned by the base library used for signature validation.
signatures[0].id	String	Signature ID
signatures[0].indication	String	Overall result of the respective signature's validation process, according to EN 319 102-1 "Table 5: Status indications of the signature validation process" The validation results of different signatures in one signature container may vary. See also validSignaturesCount and SignaturesCount fields.
signatures[0].info	Object	

Parameter	Type	Description
`signatures[0].info.bestSignatureTime`	Date	Time value that is regarded as trusted signing time, denoting the earliest time when it can be trusted by the validation application (because proven by some POE present in the signature) that a signature has existed: • in case of signature with time-mark - the producedAt value of the earliest valid time-mark (OCSP confirmation of the signer's certificate) in the signature. • in case of signature with time-stamp - the getTime value of the earliest valid signature time-stamp token in the signature.
`signatures[0].signatureFormat`	String	Format (and optionally version) of the container containing the signature(s). |
`signatures[0].signatureLevel`	String	In case of BDOC and PAdES formats: indication whether the signature is Advanced electronic Signature (AdES), AdES supported by a Qualified Certificate (AdES/QC) or a Qualified electronic Signature (QES). In case of DIGIDOC-XML 1.0..1.3 formats, empty value is used as the signature level is not checked by the JDigiDoc base library that is used for validation.
`signatures[0].signatureScopes`	Array	Contains information of the original data that is covered by the signature.
`signatures[0].signedBy`	String	Signers name and identification number
`signatures[0].subIndication`	String	Additional subindication in case of failed or indeterminate validation result, according to EN 319 102-1 "Table 6: Validation Report Structure and Semantics" |
`signatures[0].warnings`	Array	Block of validation warnings that do not affect the overall validation result. Known warning situations (according to http://id.ee/public/SK-JDD-PRG-GUIDE.pdf, chap. 5.2.4.1): • BDOC (and PAdES?): Weaker digest method (SHA-1) has been used than recommended when calculating either or element's digest value. • DIGIDOC-XML 1.0-1.3: `DataFile` element's `xmlns` attribute is missing. • DIGIDOC-XML 1.0-1.3: `<IssuerSerial><X509IssuerName>` and/or `<IssuerSerial><X509IssuerSerial>` element's `xmlns` attribute is missing. |

9.1.3.1 Sample response


```
{
    "policy": {
        "policyVersion": "1.0",
        "policyName": "SiVa validation policy",
        "policyDescription": "SiVa validation policy",
        "policyUrl": "http://open-eid.github.io/SiVa/siva/appendix/validation_policy/"
    },
    "signaturesCount": 1,
    "validSignaturesCount": 1,
    "validationTime": "2016-07-28T14:41:45Z",
    "documentName": "sample.bdoc",
    "signatures": [
        {
            "claimedSigningTime": "2016-05-11T10:17:57Z",
            "errors": [],
```

```
            "id": "S0",
            "indication": "TOTAL-PASSED",
            "info": {
                "bestSignatureTime": "2016-05-11T10:18:06Z"
            },
            "signatureFormat": "XAdES_BASELINE_LT_TM",
            "signatureLevel": "QES",
            "signatureScopes": [],
            "signedBy": "NURM,AARE,38211015222",
            "subIndication": "",
            "warnings": []
        }
    ]
}
```


9.2 SOAP validation service

9.2.1 SOAP Request

Parameter	Type	Mandatory	Description
`Document`	String	•	Base64 encoded string of digitally signed file
`Filename`	String	•	Filename of the digitally signed file (i.e `sample.bdoc`
`DocumentType`	String	•	Validation service to use for validation
`SignaturePolicy`	String	•	If You want to use alternative validation policy. Then You `signaturePolicy` for that.

9.2.2 SOAP Response

!!! development Need to add response description

10 Use cases

10.1 Digitally signed document validation process

Digitally signed document validation process shows how SiVa chooses validation service and possible output of validation process.

User of SiVa system provides digitally signed document file in form of Base64 encoded string. The validation of file and validation policy is handled by validation services underlying libraries.

- In case of PDF file it will be DSS
- For BDOC and DDOC files we will use DigiDoc4J or when required jDigiDoc
- And for X-Road signatures we will use X-road signature validation utility

We will log following failure cases: When file upload fails (request started but was not completed successfully) When request validation (JSON or SOAP) fails When user authentication fails - **not shown in diagram above** When signature validation fails – **not shown in diagram above** When increasing of request count fails – **not shown in diagram above**

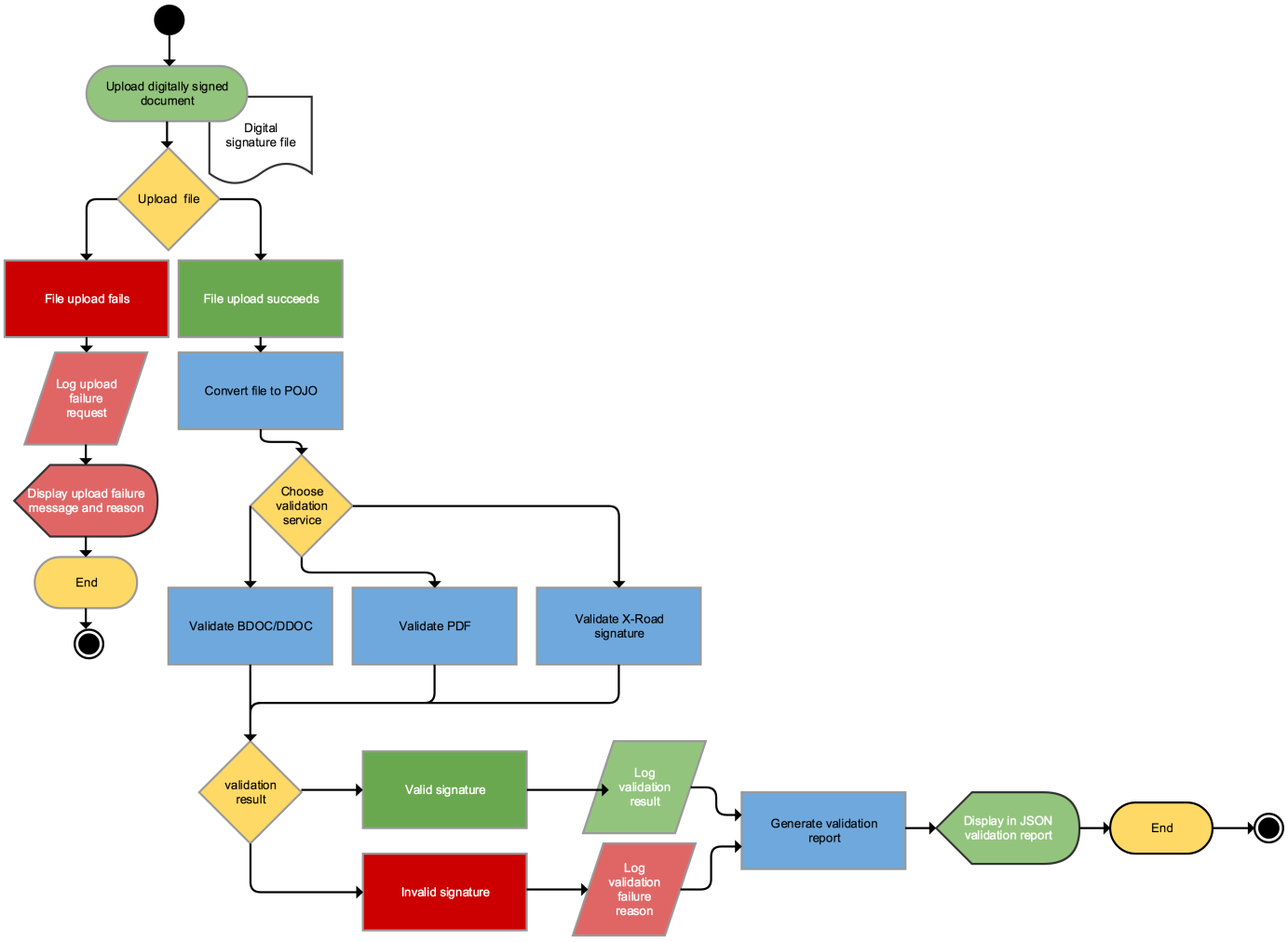

Figure 3: BDOC validation process

10.2 Certificate loading process

All validation services require certificates to validate digitally signed documents. Below process shows how certificates are loaded into validation service. Loading process is done separably for each validation service.

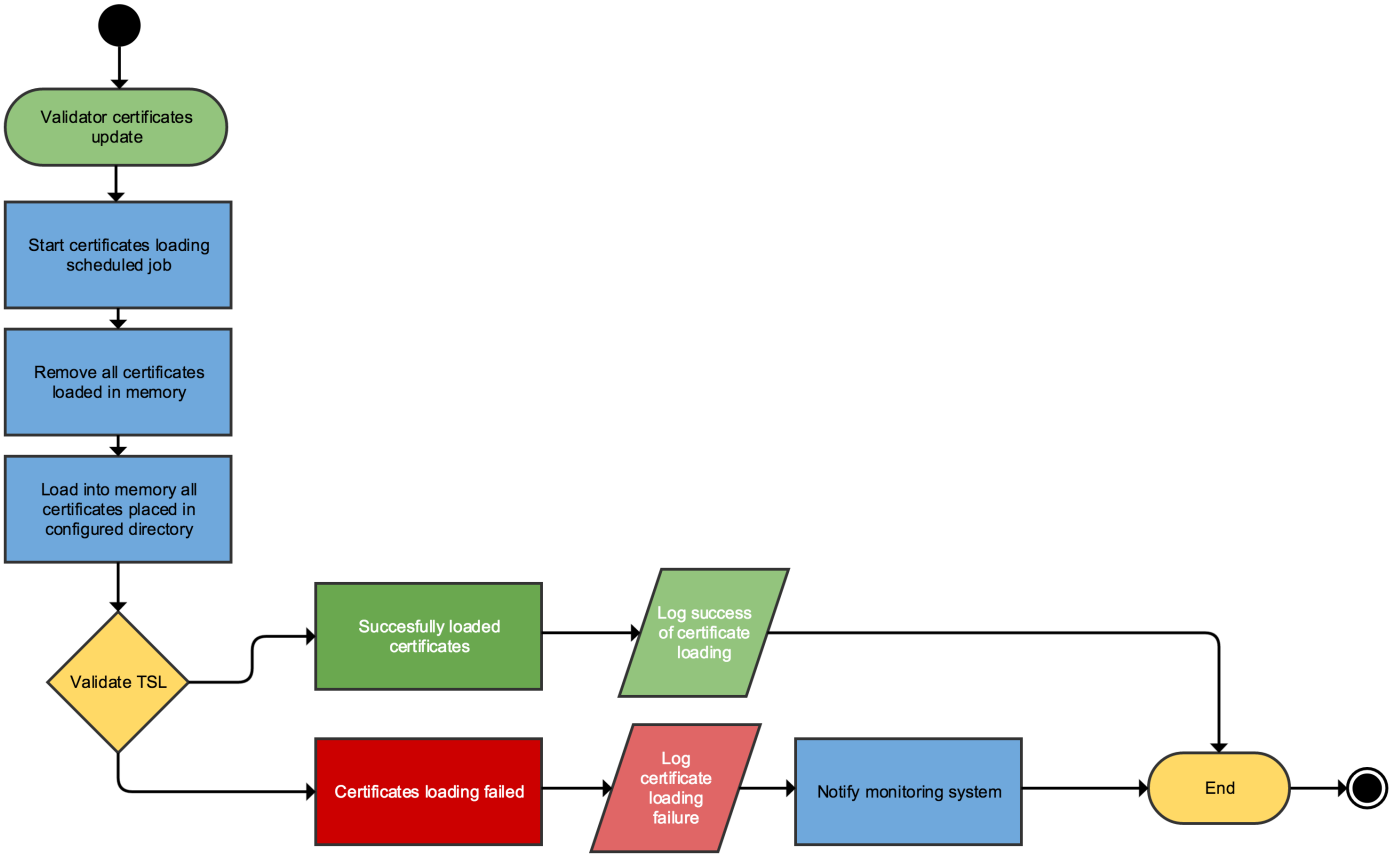

Figure 4: Certificate Loading process

Certificate loading process is scheduled cron job inside each validation service to update currently in memory loaded certificates.

This process should run after TSL loader has completed updating SiVa local copy of certificates.

10.3 X-Road 6 security server SOAP request process

X-Road validation process is brought out because we skip authentication process for X-Road security server interface and and use XML SOAP as input source.

Validation of SOAP request XML is done in the SiVa web application module. Document validation process is described in detail in Digitally signed document validation process Validation report output id described in Interface description

10.4 Authenticate JSON REST API user

Validation of JSON request is done in SiVA web application module Document validation process is described in detail in Digitally signed document validation process Validation report output id described in Interface description

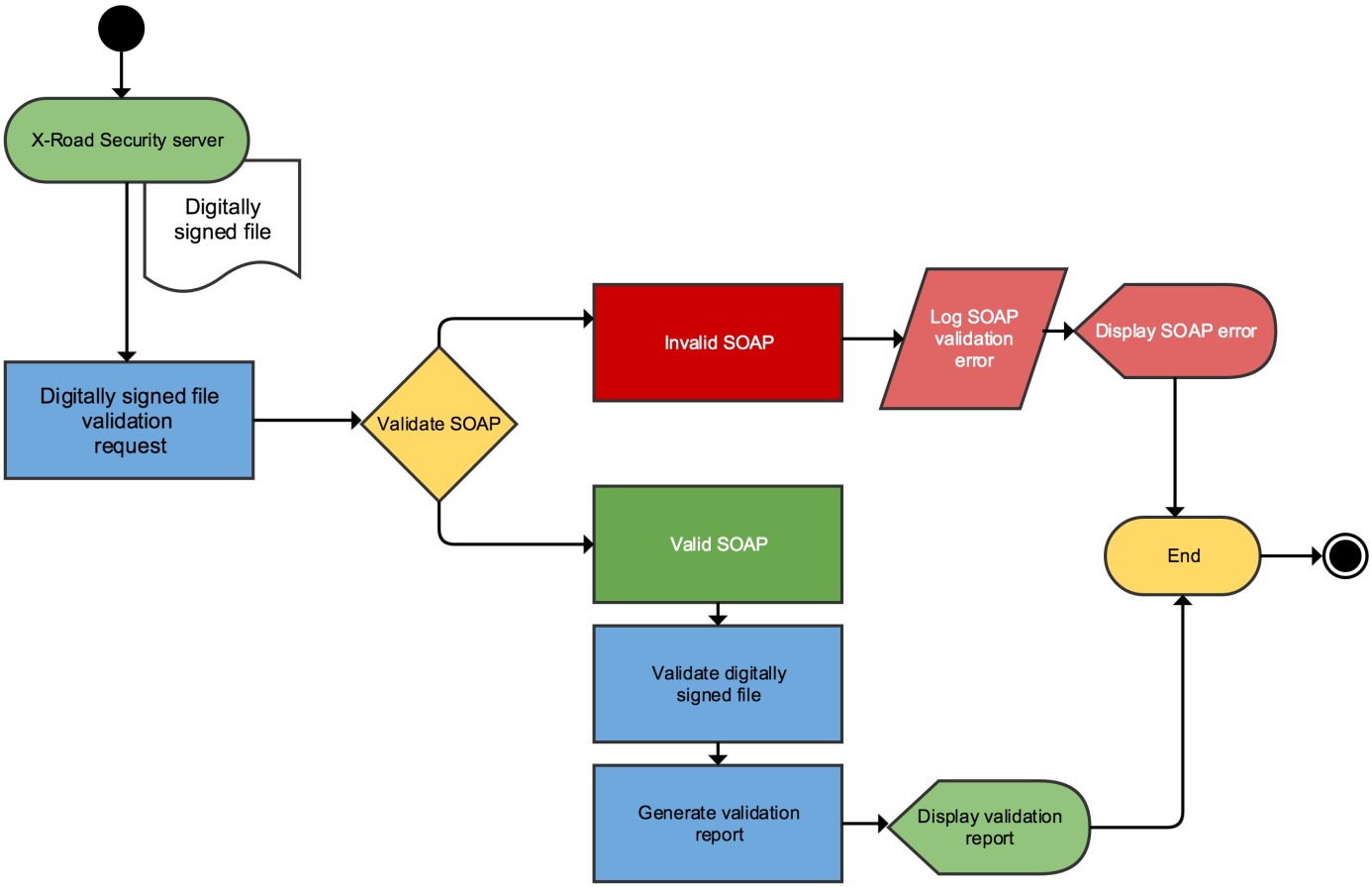

Figure 5: X-Road SOAP validation request

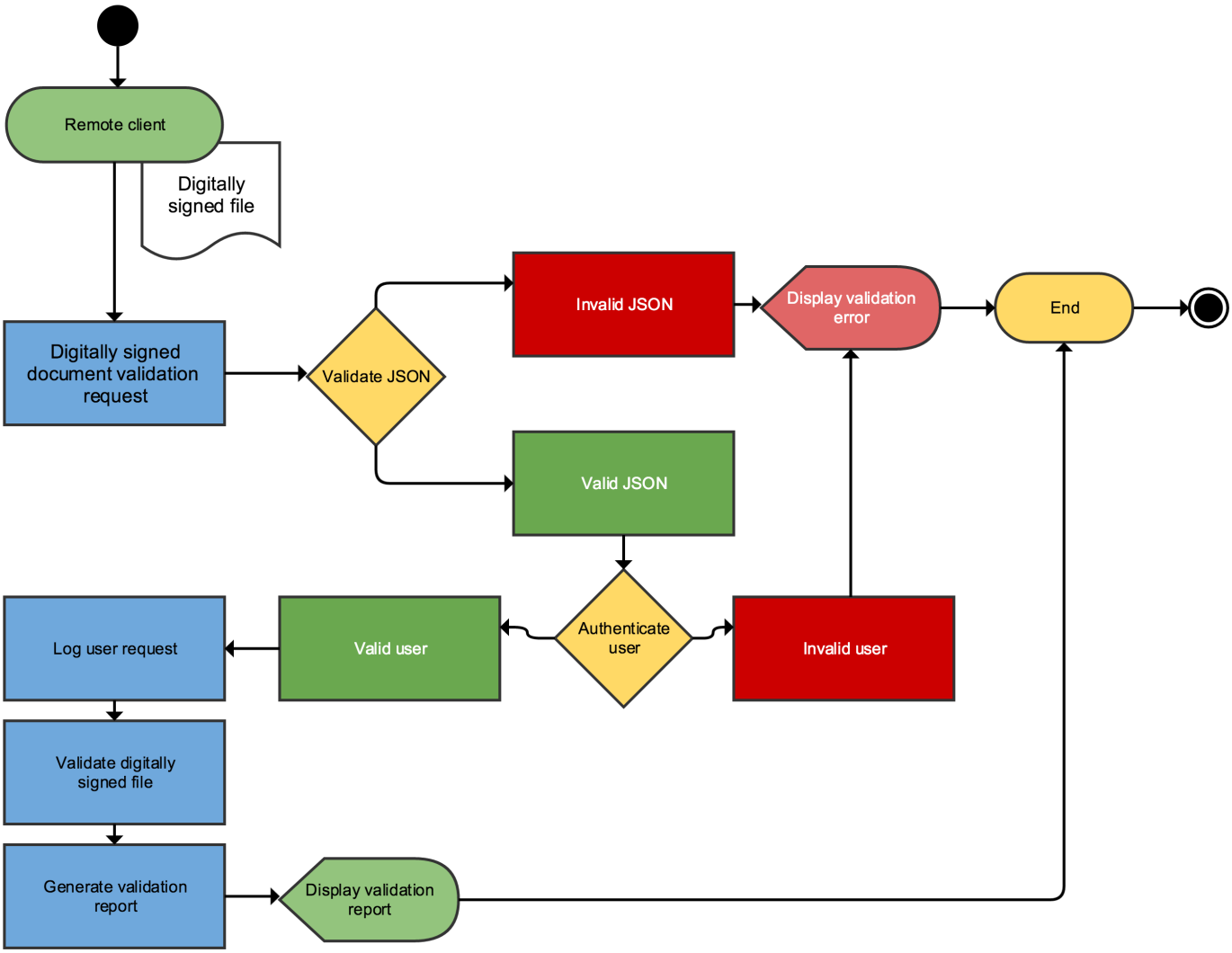

Figure 6: JSON REST validation request

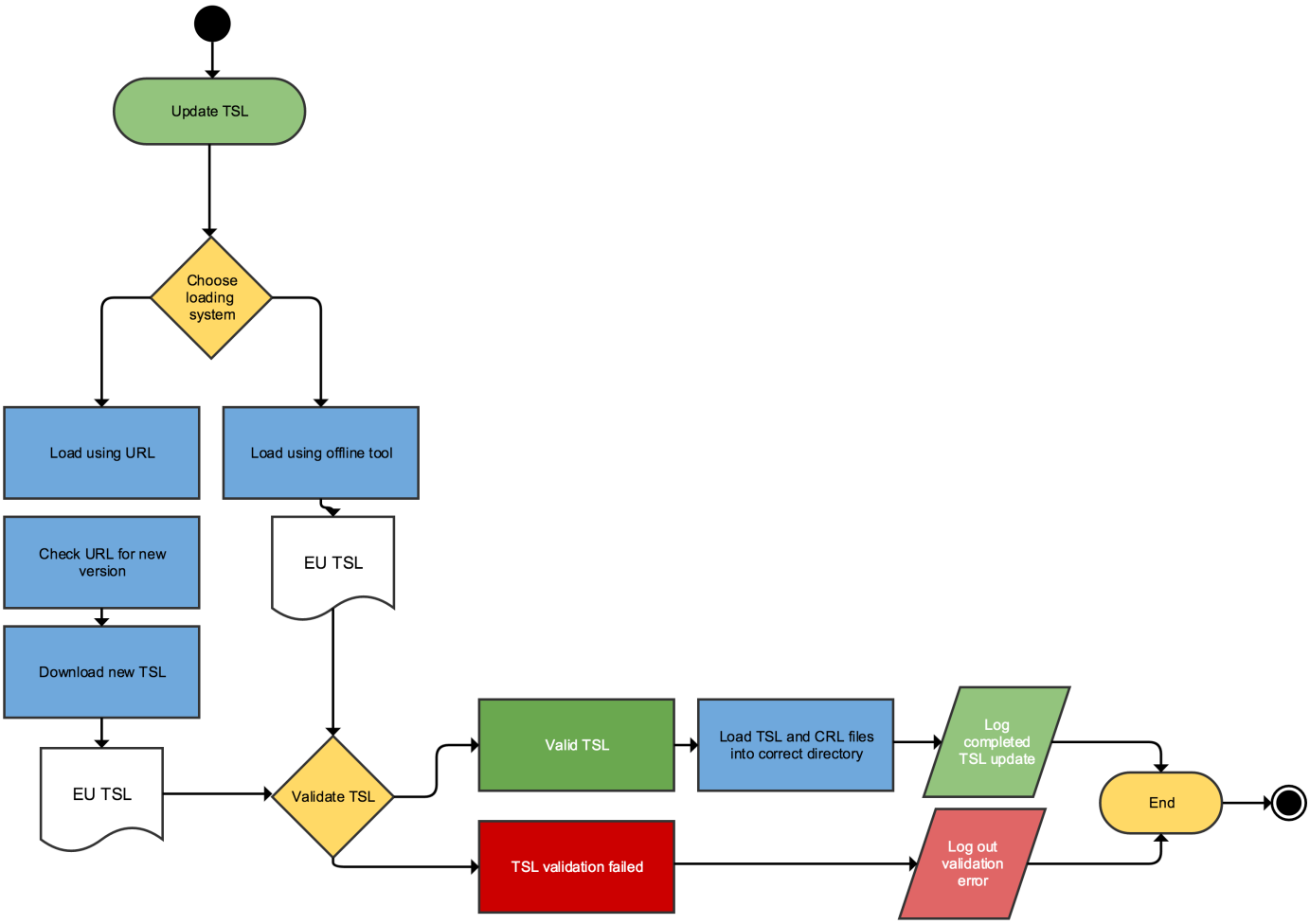

Figure 7: TSL loading process

10.5 TSL loading use case

11 Deployment

11.1 System requirements

Following are minimum requirements to build and deploy SiVa validation web service:

- Java 8 or above Oracle JVM is supported
- Git version control system version 1.8 or above is recommended
- Minimum 2 GB of RAM. Recommended at least 4 GB of RAM
- Minimum 1 processor core
- Open internet connection
- 1GB of free disk space
- Supported operating system is Ubuntu 14.04 LTS

11.2 Building SiVa validation web service on Ubuntu 16.04

First we need to install Git and Java SDK 8 by issuing below commands:

```
sudo apt-get update
sudo apt-get install git -y
sudo apt-get install default-jdk -y
```

Next we need to clone the SiVa Github repository:

```
git clone https://github.com/open-eid/SiVa.git --branch develop
```

Final step is building the SiVa project using Maven Wrapper

```
cd SiVa
./mvnw install
```

!!! note The build can take up to **30 minutes** because there are lot of tests that will be run through and downloading of the required dependencies

To verify that SiVa project built successfully look for `BUILD SUCCESS` in build compilation output last lines. The last lines of build output should look very similar to below image:

```
[INFO] Reactor Summary:
[INFO]
[INFO] SiVa Digitally signed documents validation service . SUCCESS [ 25.258 s]
[INFO] validation-services-parent ..... SUCCESS [  0.479 s]
[INFO] validation-commons ..... SUCCESS [01:45 min]
[INFO] tsl-loader ..... SUCCESS [ 16.507 s]
[INFO] PDF Validation Service ..... SUCCESS [ 42.263 s]
[INFO] BDOC Validation Service ..... SUCCESS [ 58.864 s]
[INFO] DDOC Validation Service ..... SUCCESS [  9.929 s]
[INFO] xroad-validation-service ..... SUCCESS [  5.664 s]
[INFO] SIVa webapp and other core modules ..... SUCCESS [  0.315 s]
[INFO] SiVa validation service proxy ..... SUCCESS [ 43.098 s]
[INFO] siva-webapp ..... SUCCESS [04:06 min]
[INFO] SiVa Sample Web application ..... SUCCESS [04:31 min]
[INFO] SiVa Web Service integration tests ..... SUCCESS [03:41 min]
[INFO] siva-distribution ..... SUCCESS [ 56.941 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 18:30 min
[INFO] Finished at: 2016-07-22T13:40:31+00:00
[INFO] Final Memory: 80M/250M
[INFO] ------------------------------------------------------------------------
```

11.3 Setting up systemd service

Maven build generates executable JAR files. This means web container and all its dependencies are package inside single JAR file. It makes a lot easier to deploy it into servers.

Easiest option to setup SiVa is as `systemd` service in Ubuntu servers.

For that we first need to create service file:

```
vim siva-webapp.service
```

Inside it we need to paste below text. You need and can change few things in service setup file.

- First you **must not** run service as `root`. So it's strongly recommended to change line `User=root`
- Second You can change Java JVM options by modifying the `JAVA_OPTS` inside the `siva-webapp.service` file.
- Also You can change the SiVa application configuration options by modifying `RUN_ARGS` section in file

```
[Unit]
Description=siva-webapp
After=syslog.target

[Service]
User=root
ExecStart=/var/apps/siva-webapp.jar
Environment=JAVA_OPTS=-Xmx320m RUN_ARGS=--server.port=80
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target
```

Save and close the `siva-webapp.service` file. Next we need to move `siva-webapp-2.0.2-SNAPSHOT.jar` into newly created `/var/apps` directory and rename to JAR file to `siva-webapp.jar`. match

!!! note The copied JAR filename must match option `ExecStart` in `siva-webaapp.servcie` file

```
sudo mkdir /var/apps
sudo cp siva-parent/siva-webapp/target/executable/siva-webapp-2.0.2-SNAPSHOT.jar /var/apps/siva-webapp.jar
```

Next we need to copy the `siva-webapp.service` file into `/lib/systemd/system` directory. Then we are ready to start the `siva-webapp` service.

```
sudo cp siva-webapp.service /lib/systemd/system
sudo systemctl start siva-webapp
```

Final step of setting up the `siva-webapp` service is to verify that service started correctly by issuing below command.

```
systemctl status siva-webapp
```

It should print out similar to below picture:

```
 siva-webapp.service - siva-webapp
   Loaded: loaded (/lib/systemd/system/siva-webapp.service; disabled; vendor preset: enabled)
   Active: active (running) since Thu 2016-07-21 08:48:14 EDT; 1 day 2h ago
 Main PID: 15965 (siva-webapp.jar)
    Tasks: 34
   Memory: 429.6M
      CPU: 2min 5.721s
   CGroup: /system.slice/siva-webapp.service
           15965 /bin/bash /var/apps/stage/siva-webapp.jar
         15982 /usr/bin/java -Dsun.misc.URLClassPath.disableJarChecking=true -Xmx320m -jar /var/apps/stage/siva-w

Jul 20 03:00:01 siva siva-webapp.jar[15965]:      at eu.europa.esig.dss.tsl.service.TSLParser.getTslModel(TSLPar
Jul 20 03:00:01 siva siva-webapp.jar[15965]:      at eu.europa.esig.dss.tsl.service.TSLParser.call(TSLParser.jav
Jul 20 03:00:01 siva siva-webapp.jar[15965]:          ... 5 common frames omitted
Jul 20 03:00:01 siva siva-webapp.jar[15965]: 20.07.2016 03:00:01.450 INFO  [pool-3-thread-1] [e.e.e.dss.tsl.servic
Jul 20 03:00:01 siva siva-webapp.jar[15965]: 20.07.2016 03:00:01.450 INFO  [pool-3-thread-1] [e.e.e.dss.tsl.servic
```

11.4 Installing HTTPIE

`httpie` is more user friendly version of `curl` and we will use to verify that SiVa was installed and started correctly on our server.

11.4.1 Ubuntu 16.04

On Ubuntu You can install it using `apt` package manager:

```
sudo apt-get install -y httpie
```

11.4.2 Mac OS X

On Mac it's strongly recommended to install it using package manager like Homebrew by issuing belwo command:

```
brew install httpie
```

11.4.3 Windows

On Windows there is no prebuilt package that can be installed but `httpie` installation instruction in Scott Hanselmans blog post

11.4.4 Cross platform

Alternative option if You have Python and its package manager `pip` installed. Then You can issue below command:

```
pip install httpie
```

11.5 Verify digitally signed file

After You have successfully installed HTTPIE. Then we need to download sample JSON request file with below command

```
http --download https://raw.githubusercontent.com/open-eid/SiVa/develop/build-helpers/bdoc_pass.json
```

After successful download issue below command in same directory where You downloaded the file using the above command.

```
http POST http://10.211.55.9:8080/validate < bdoc_pass.json
```

Output of this command should look like below screenshot. Look for `signatureCount` and `validSignatureCount` they **must** be equal.

12 Logging

Logging functionality is handled by the **SLF4J** logging facade and on top of it the **Logback** framework is used. As a result, logging can be configured via the standard Logback configuration file. By default, logging works on the `INFO` level and logs are directed to the system console as well as a log file.

The logback xml configuration file can be found at:

`pdfValidator/pdf-validator-webapp/src/main/resources/logback.xml`

and when compiled the file will reside at

`WEB-INF/classes/logback.xml`

within the packaged war. There is also a possibility to set the location of the default configuration file with a system property `logback.configurationFile` as a JVM argument. The value of this property can be a URL, a resource on the class path or a path to a file external to the application.

```
└─$ http POST http://10.211.55.9:8080/validate < build-helpers/bdoc_pass.json
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: application/json;charset=UTF-8
Date: Fri, 22 Jul 2016 16:12:35 GMT
Transfer-Encoding: chunked

{
    "documentName": "Valid_IDCard_MobID_signatures.bdoc",
    "policy": {
        "policyDescription": "SiVa validation policy",
        "policyName": "SiVa validation policy",
        "policyUrl": "http://open-eid.github.io/SiVa/siva/appendix/validation_policy/",
        "policyVersion": "1.0"
    },
    "signatures": [
        {
            "claimedSigningTime": "2016-05-11T10:17:57Z",
            "errors": [],
            "id": "S0",
            "indication": "TOTAL-PASSED",
            "info": {
                "bestSignatureTime": "2016-05-11T10:18:06Z"
            },
            "signatureFormat": "XAdES_BASELINE_LT_TM",
            "signatureLevel": "QES",
            "signatureScopes": [],
            "signedBy": "NURM,AARE,38211015222",
            "subIndication": "",
            "warnings": []
        },
        {
            "claimedSigningTime": "2016-05-11T10:19:09Z",
            "errors": [],
            "id": "S1",
            "indication": "TOTAL-PASSED",
            "info": {
                "bestSignatureTime": ""
            },
            "signatureFormat": "XAdES_BASELINE_LT",
            "signatureLevel": "QES",
            "signatureScopes": [],
            "signedBy": "PIIGLI,MADIS,38603195225",
            "subIndication": "",
            "warnings": []
        }
    ],
    "signaturesCount": 2,
    "validSignaturesCount": 2,
    "validationTime": "2016-07-22T16:12:35Z"
}
```

Figure 8: HTTPIE output validation

```
java -Dlogback.configurationFile=/path/to/config.xml
```

In this configuration file there are three appenders: `STDOUT` (logs to standard output), `FILE` (logs to a file) and `SYSLOG` (logs to syslog server over the network). To disable certain appender from logging, commenting out its `appender-ref` is sufficient, but it is *recommended* that the appender itself should also be commented out. For example to disable `SYSLOG` appender (which is the default configuration), then one can use following configuration:

```
<!--
<appender name="SYSLOG" class="ch.qos.logback.classic.net.SyslogAppender">
    <syslogHost>enter\_ip\_or\_hostname\_here</syslogHost>
    <port>514</port>
    <facility>USER</facility>
    <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
        <level>INFO</level>
    </filter>
    <suffixPattern>%-5level %logger{0}:%L \[%thread\] - %msg</suffixPattern>
</appender>
-->

<root level="DEBUG">
<appender-ref ref="STDOUT"/>
<appender-ref ref="FILE"/>
<!--<appender-ref ref="SYSLOG"/>-->

</root>
```

Logback configuration manual: http://logback.qos.ch/manual/

12.1 STDOUT appender

- Default the log level is set to DEBUG
- Appender output pattern is: `%d{HH:mm:ss.SSS} %-5level %logger{0}:%L [%thread] - %msg%n`

12.2 FILE appender

- Default log level is set to `INFO`
- uses RollingFileAppender configured with `TimeBasedRollingPolicy`. Current configuration makes a seperate logfile for each day and each file is kept for *30 days*.

PS! keep in mind when using relative destination file path, then the path is added at the end of the currently working directory, i.e. where the application was started. (Current day's logfile path: `logs/pdf-validator-webapp.log`, prievious days pattern:

`logs/pdf-validator-webapp.%d{yyyy-MM-dd}.log)`

- Appender output pattern is: `%d{HH:mm:ss.SSS} %-5level %logger{0}:%L \[%thread\] - %msg%n`

  `-Dlogback.configurationFile=config.xml`

12.3 SYSLOG appender

- Default log level is set to `INFO`
- Target's ip/hostname and port are configurable
- Syslog messsages' severity is configurable
- Syslog messages' payload's timestamp and hostname part are created implicitly and the suffixpattern is: `%-5level %logger{0}:%L \[%thread\] - %msg`

13 QA Strategy

13.1 Introduction

The goal of this document is to give general overview of the used infrastructure, processes, schedule and actions to ensure good quality delivery. The document describes activities in the whole software development process. Analysis, development and testing are separated for the sake of structure and transparency although they are integral parts of the development cycle.

13.2 Environments and infrastructure

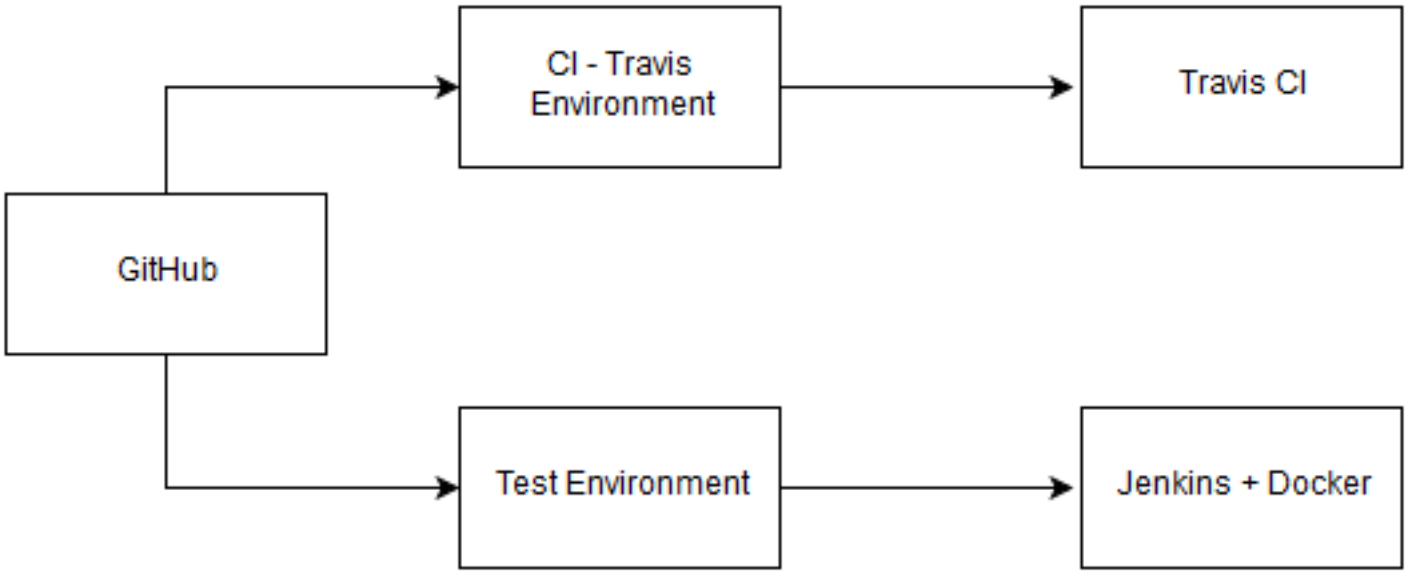

Figure 9: Enviroment

There are different test environments for quality assurance, depending on the nature and aim.

1. TravisCI environment for public - Platform: Linux
2. Test environment for test and performance test - Platform: Linux

Instructions how to set up test enviroment and run tests together with more info can be found in SiVa GitHub page

System requirements:

- At least 2GB of RAM on machine where the build is executed
- Minimum required Java version is Java 8
- SiVa project provided Maven Wrapper (./mvnw)

Tools used:

- TravisCI – is a continuous integration service used to build and test software projects hosted at GitHub
- Jenkins – is an open source continuous integration tool written in Java.
- JMeter – tool for creating and running performance tests.
- IntelliJ IDEA – is a Java integrated development environment(IDE) for developing automated tests
- Apache Tomcat - is an open source servlet container developed by the Apache Software Foundation.
- Docker – is an open-source project that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating-system-level virtualization on Linux.
- Rest-Assured - is a Java DSL(Domain-specific language) for simplifying testing of REST based Services built on top of HTTP Builder.

13.3 Analysis

Analysis will be tagged with identificators to enable cross-reference between requirements and corresponding tests. This includes both functional and non-functional requirements. Analysis of specific tasks must be done before the sprint planning

meeting. This will ensure that upcoming sprint is planned in common grounds.

Reference document [2]. and [3]. (see reference section)

13.4 Development

13.4.1 Development process

Customized iterative process similar to SCRUM is used in the development. The process consists of following elements:

- Product backlog is maintained
- Sprints are two weeks long
- Sprint planning meetings are held
- Tasks are maintained through JIRA SCRUM board
- Daily team stand-ups are held
- Tasks marked done are developed, tested and ready to be shipped
- Bi-weekly meetings are held to give status on progress and discuss open questions
- Retrospectives are held at least with two month intervals

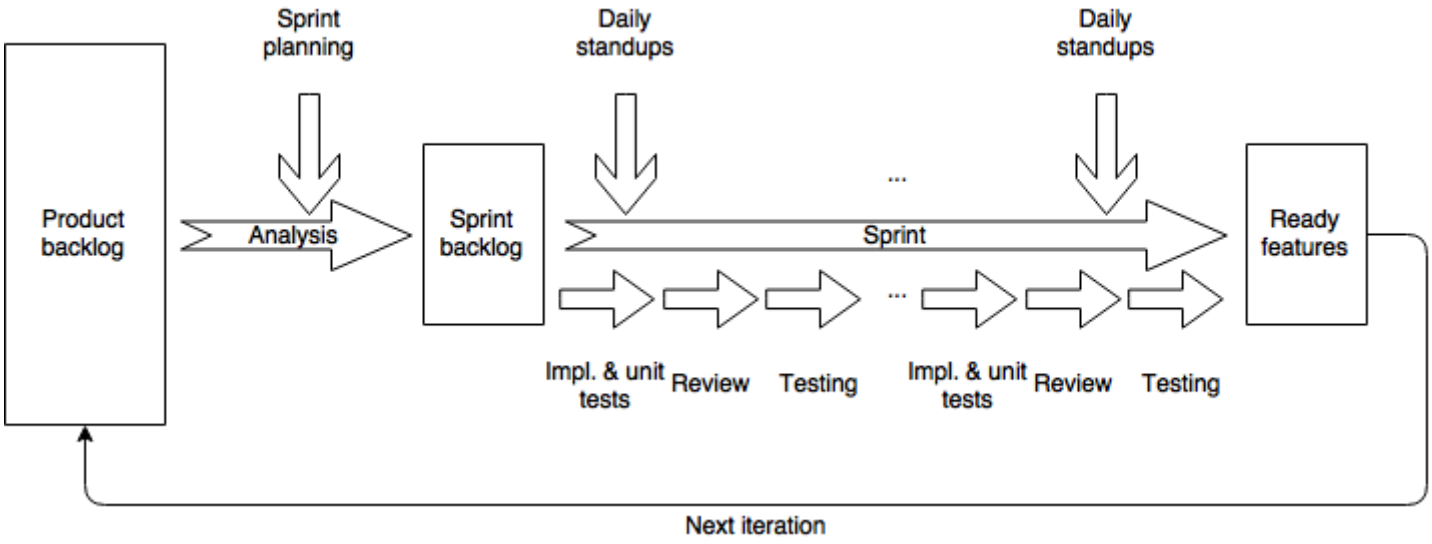

Figure 10: Development process

13.4.2 Issue lifecycle

Each ticket (defect, task, …) in JIRA SCRUM board goes through the following states that correspond the development procedure described in previous chapter.

Description of states:

- ToDo - tickets to be dealt with in current sprint (Sprint backlog).
- Analysis -ticket is being analysed. During this stage requirements are refined.
- Development - ticket is being handled (feature coding, document writing, …).
- In Review - the work done in development stage is being reviewed.
- Test - ticket is being tested.
- Done - ticket has been tested and deemed sufficient for release (Ready features)
- Closed - ticket has been released.

13.4.3 QA activities and quality criterias in the development

Process improvement

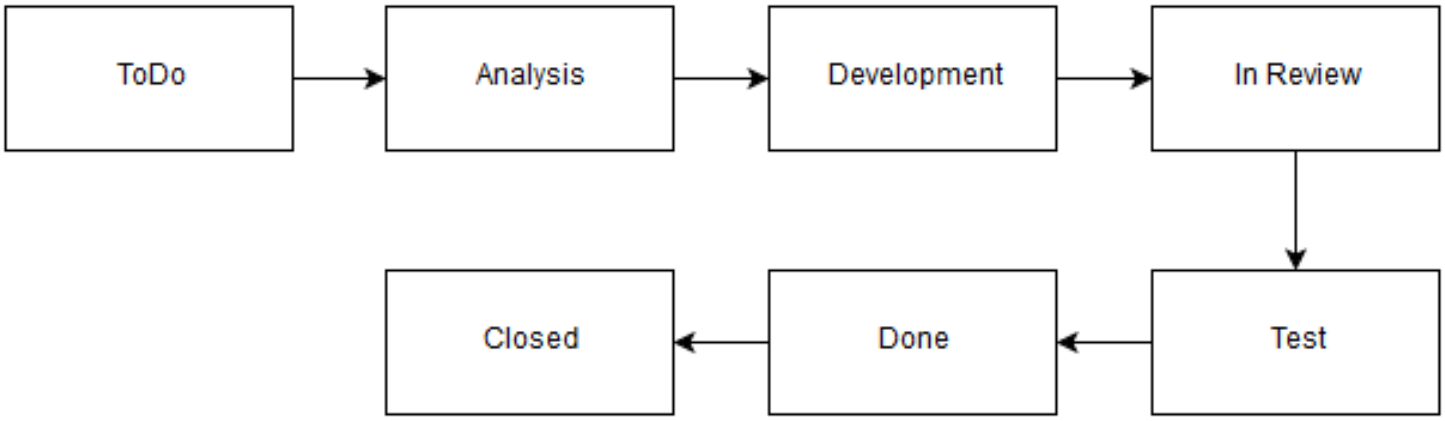

Figure 11: Issue lifecycle

The development process is constantly monitored and adjusted to the changing situations. Retrospective meetings for process feedback are held.

Unit tests

It is responsibility of developer to write, maintain and execute the unit tests on developed features. The code must compile and pass unit tests before it is allowed to be submitted to the code repository. The code of the unit tests is integral part of the source code and is maintained on the same principles.

Unit tests are also automatically executed on each build, if the unit tests do not pass further test execution is stopped.

Static testing/code reviews

All changes (including changes in unit test code) are reviewed by another development team member using GitHub. The code must pass review before it is submitted to testing.

SonarLint is used to validate code automatically. It integrates both suggested tools mentioned in reference document [3]. (see reference section)

13.5 Testing

13.5.1 Approach and schedule

Testing follows the principles described in reference document [1]

The goal is to automate as much of the testing process as possible, however some aspects of the testing will be carried out manually.

As the development is carried out validation module by validation module, testing will follow the same principle. The tests will be developed in iterations for each validation module.

After each iteration test cases, test automation code and test results will be available through GitHub.

The main focus on testing is on different type of input data, the tests themselves will be based on the same principle for all the modules.

The schedule of testing can be seen on the image below.

13.5.2 Testing process

All automatic tests, except load tests will follow the same execution process. The tests are ran automatically and all the actions are triggered by Jenkins. The full set of tests (unit, integration and system) will be ran on every test execution.

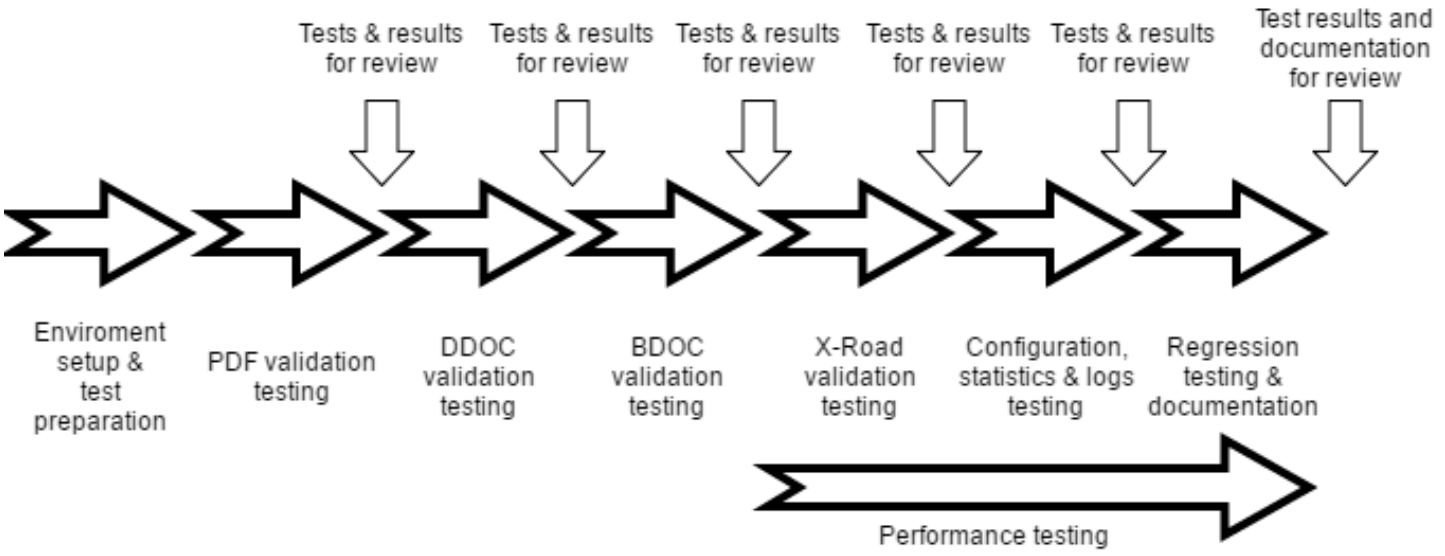

Figure 12: Testing schedule

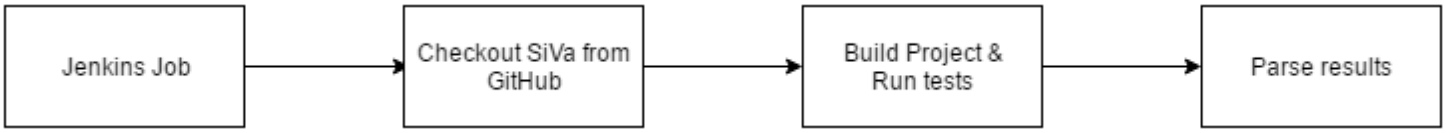

Figure 13: Testing process

13.5.3 Test case management

Test cases are handled as integral part of test automation code. The same applies on manual tests, in manual test cases some portion of automation may be used to execute the tests but the results are verified manually. All the test cases and test code will be maintained in the GitHub.

Test cases are developed and maintained together with test automation code by the QA specialist for Integration and System Testing.

Following elements will be present in test cases:

- TestCaseID: unique ID that makes possible to identify specific test case
- RequirementID: ID of the requirement that is tested
- Title: Description of the test
- Expected Result: expected outcome of the test (pass criteria for the test)
- File: input test file that is used

Test case sample

Automatic and manual test cases will have the same description principles (shown below).

```
/***
 *
 * TestCaseID: PDF-BaselineProfileTests-1
 *
 * TestType: Automated/Manual
 *
 * RequirementID: SiVa-I-8.1
 *
 * Title: The PDF-file has got signatures of two different profiles - Signing profiles PAdES Baseline-B and PA
 *
 * Expected Result: Signature profile Baseline - LTA should pass and signature baseline-b should fail
 *
 * File: hellopades-lt-b.pdf
```

```
 *
 ***/
@Test

public void baselineProfileLTDocumentShouldPass(){
    String filename = "hellopades-lt-b.pdf";
    String encodeFile = Base64.encodeBase64String(readFile(filename));
    String document = encodeFile;
    ValidationRequest validationRequest = new ValidationRequest();
    validationRequest.setDocumentType(parseFileExtension(filename.substring(filename.lastIndexOf(".") + 1)));
    validationRequest.setDocument(document);
    restTemplate.postForObject(sivaBaseUrl, validationRequest, String.class);
}
```

13.5.4 Defect management

All found defects will be reported and handled in Jira. The defect report will follow similar lifecycle in JIRA SCRUM board as tasks.

The report will have following elements:

- Title: Short, precise description of the problem
- Details: Type, Priority, Sprint, Epic Link, Fix Version/s
- Description:
  - **Steps to reproduce bug** - sequence for reproduction, link to test case if applicable.
  - **Expected behavior** - expected outcome of the test sequence.
  - **Actual behavior** - actual result of the test sequence. The description should be thorough enough to enable the debugging of the problem and to give objective base for severity assessment.
  - **File attachments** - Test files, logs, images, …

13.5.5 Test levels

Integration testing

The scope of the tests is illustrated on the image below. The goal is to test the SiVA application API (both X-Road and REST/JSON) and to test the independent module capability for validation of specific type of file (DDOC, BDOC, PDF or X-Road signature). Both valid and invalid inputs are tested. More info about testing specifics can be found in Test Plan Integration testing section.

System testing

The scope of the tests is illustrated on the image below. The goal of the test is to test the entire length of signature validation process and to test supportive functions. In addition REST/JSON Demo application is tested. X-Road application is used to test the flow, but application itself is not tested. More info about testing specifics can be found in Test Plan System testing section.

Regression testing

Regression testing will consist of two parts: Running all automated tests (unit, integration and system tests) Manual testing of the areas that are not covered by automatic tests based on the regression test checklist

13.5.6 Additional testing activities

Performance testing

Performance testing will be carried out using JMeter on Test Environment. The testing will be carried out for each type of validation service.

More info about performance testing specifics can be found in Test Plan Performance testing section.

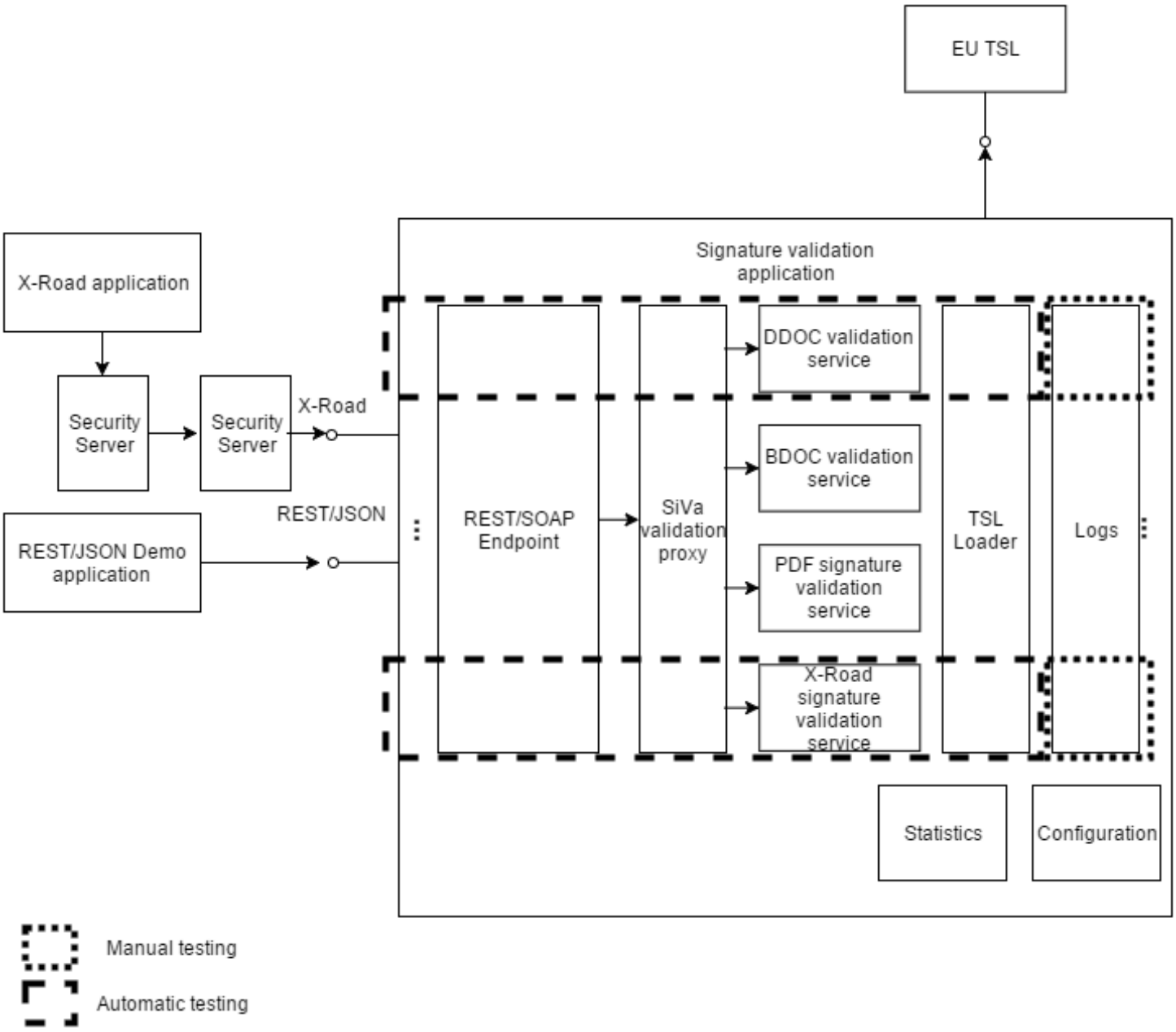

Figure 14: Integration testing

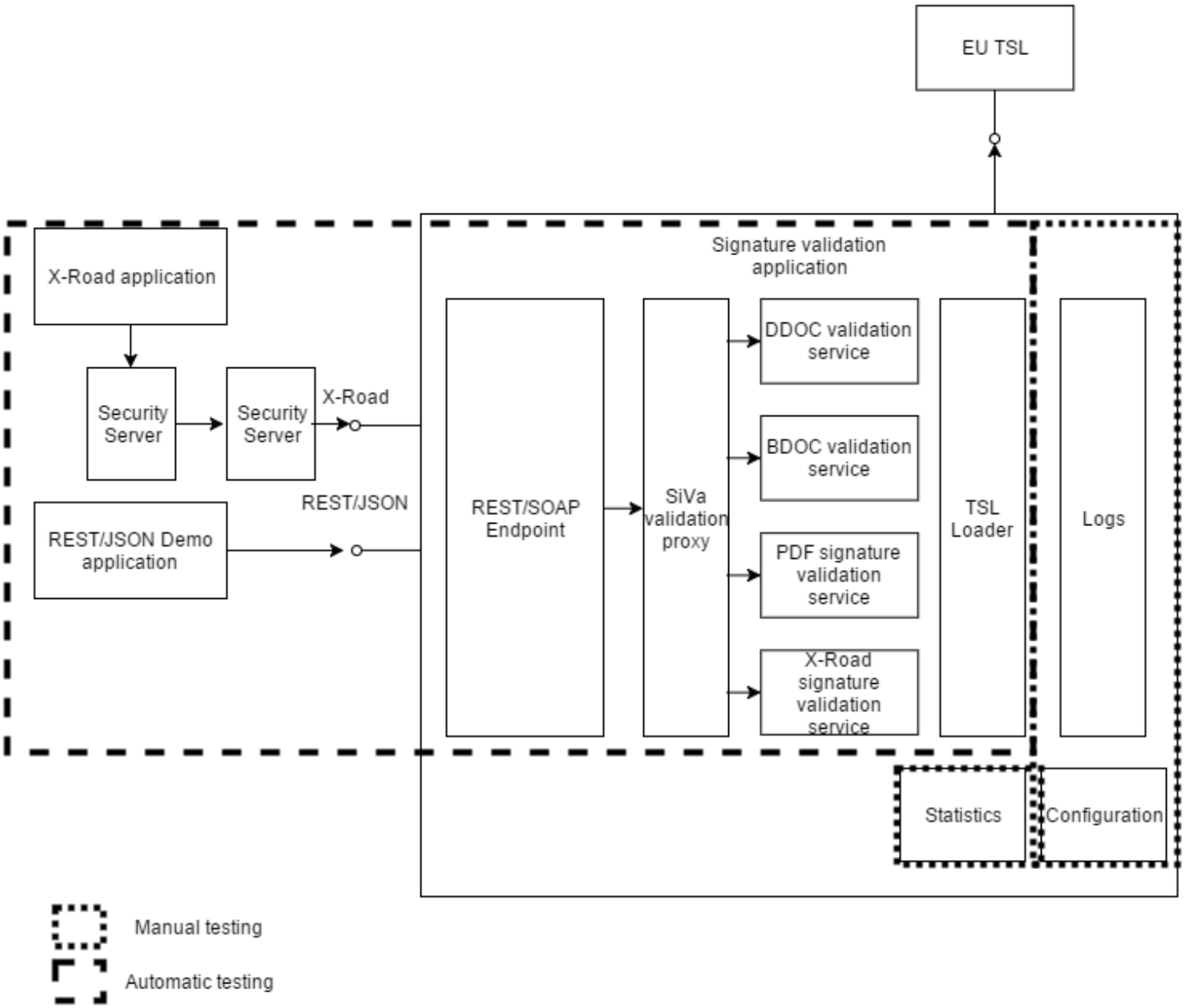

Figure 15: System testing

13.6 References

!!! development This section will be moved to general reference page

- [1] Lisa_6_Osa_I_SiVa_Testimise_korraldus.pdf
- [2] Lisa_4_Osa_I_SiVa_Valideerimisteenuse_analuus MUUDETUD.pdf
- [3] Lisa_3_RIA_Mittefunktsionaalsed_nouded.pdf
- [4] OWASP Testing Guide 4.0

14 Test Plan

!!! development ** Unclear items that are not yet present in the plan **

```
* Testing of logs
* Testing of user statistics
* Configuration/administration of the service
* SOAP API testing
* System Testing of X-Road service
```

This is living document and will be constantly updated as the project evolves. The aim of the document is to describe what is tested during SiVa Web Apllication development.

14.1 Integration Test introduction

Overview of the SiVa (Signature Validation) web service can be found in the Overview section of the document.

This section of the document gives overview of Integration Testing carried out on SiVa web service.

Integration testing is using RestAssured library v2.5.0 to implement automatic checks for REST/SOAP based tests.

The testing of the SiVa web service is divided into sections based on the software architecture and functionalities provided to the users. The sections are:

- REST API
- SOAP API
- DDOC signature validation
- BDOC signature validation
- PDF signature validation
- X-Road ASICE signature validation

The goal is to focus testing on functionalities implemented in SiVa application. Functionalities provided by Validation libraries are not explicitly tested.

14.2 Testing of REST API

The goal of the REST API testing is to check that the API is accepting the requests based on the specification and the output result (Validation Report) is in correct format and has all the required elements.

14.2.1 Validation REST API tests

Following areas are tested on input:

- Wrong (not accepted) values in input parameters
- Empty values in input paramters
- Too many parameters
- Too few parameters
- Inconsistencies on stated parameters and actual data (wrong document type)
- Case insensitivity on parameter names
- Empty request
- Simultaneous validation requests

In all of the negative cases correctness of returned error message is checked.

Specific test cases and input files can be found in:

- Appendix 5 - ValidationRequestTests.java
- Appendix 5 - DocumentFormatTests.java

14.2.2 Validation Report tests

SiVa web service returns uniform Validation Report on all the supported document types. This also includes correct document types without actual signature (for example PDF document without signature).

Following areas are tested on output (Validation Report):

- JSON structure on DDOC, BDOC, PDF and ASICE document types
- Presence of the mandatory elements on DDOC, BDOC, PDF and ASICE document types
- Presence of optional elements on DDOC, BDOC, PDF and ASICE document types
- JSON structure on containers without signatures

What is not tested:

- Correctness of the values in the report is not in scope on these tests

Specific test cases and input files can be found in:

- Appendix 5 - ValidationReportJsonStructureVerification.java

14.3 Testing of SOAP API

!!! development Will be covered when SOAP API is implemented

14.4 Testing of DDOC signature validation

The goal of the DDOC signature validation testing is to check that the validation results given by JDigiDoc library are properly presented in validation report.

The testing of DDOC signatures consists of following main cases:

- Containers with valid signature(s) are validated.
- Containers with invalid signature(s) or no signature are validated
- Containers sizes near maximum are validated
- Containers with DDOC v1.0 - 1.3 are validated

Specific test cases and input files can be found in:

- Appendix 5 - DdocValidationFail.java
- Appendix 5 - DdocValidationPass.java
- Appendix 5 - LargeFileTests.java

What is not tested:

- Verification of different causes in container for invalid result is out of scope.

!!! development Insert links to JDOC repos where these were tested.

14.5 Testing of BDOC signature validation

The goal of the BDOC signature validation testing is to check that the validation results given by DigiDoc4J library are properly presented in validation report.

The testing of BDOC signatures consists of following main cases:

- Containers with valid signature(s) are validated (how many signatures are acceptable?)
- Containers with invalid signature(s) or no signature are validated
- Containers sizes near maximum are validated

Specific test cases and input files can be found in:

- Appendix 5 - BdocValidationFail.java
- Appendix 5 - BdocValidationPass.java
- Appendix 5 - LargeFileTests.java

What is not tested:

- Verification of different causes in container for invalid result is out of scope.

!!! development Insert links where Digidoc4J repos where these were tested.

14.6 Testing of PDF signature validation

Portion of the validation rules for PDF documents are implemented in SiVa web apllication itself. Therefor different test area selection is used for PDF compared to other containers.

The testing of PDF signatures consists of following main cases:

- Containers with invalid signature(s) (different reasons for failure) are validated
- Containers with no signature are validated
- Containers sizes near maximum are validated
- Containers with different baseline profiles are validated
- Containers with serial and parallel signatures are validated
- Containers with different signature cryptocaphic algorithms are validated
- Containers with OCSP values inside and outside bounds are validated

Specific test cases and input files can be found in:

- Appendix 5 - PdfBaselineProfileTests.java
- Appendix 5 - PdfSignatureCryptographicAlgorithmTests.java
- Appendix 5 - PdfValidationFail.java
- Appendix 5 - PdfValidationPass.java
- Appendix 5 - SignatureRevocationValueTests.java
- Appendix 5 - LargeFileTests.java

14.7 Testing of X-Road ASICE signature validation

The goal of the ASICE signature validation testing is to check that the validation results given by X-Road signature validation utility are properly presented in validation report.

It is possible to validate ASICE containers both via SOAP and REST interface. The same test cases are used for both interfaces.

The testing of ASICE signatures consists of following main cases:

- Containers with valid signature(s) are validated
- Containers with invalid signature(s) and no signature are validated
- Containers sizes near maximum are validated

Specific test cases and input files can be found in Appendix 5 - Test Case Descriptions

What is not tested:

- Verification of different causes in container for invalid result is out of scope.

14.8 System Test introduction

!!! development Will be covered before System Test start

While Integration Tests were carried out automatically then System Testing will be done manually.

System testing is carried out using two access points:

- Testing through SiVa Sample Application
- Testing through X-Road Security Server

The goal is to test the whole validation process.

14.9 Additional features

- “Offline” mode of SiVa web apllication
- Configuration of validation policy

14.10 SiVa Sample Application tests

In addition to testing the service as such, SiVa Sample Application itself is tested. The main cases are:

- Cross browser usage (IE, Edge, Chrome, Firefox and Safari)
- File upload (different sizes, suported and unsupported file types)
- Displayment of Validation Report

14.11 Performance Test introduction

Performance testing will be carried out on following environments:

- Nortal Load Test (processor: memory:)

Jmeter v2.13 is used to carry out the testing.

The goal is to measure throughput-latency of the service with different file types and sizes. The performance testing is carried out on REST interface. SOAP interface is used only for one testrun for comparison between SOAP and REST interface. It is assessed that the interface itself does not have considerable impact on throughput or latency compared to the validation process.

Following cases will be covered on all supported file types (BDOC, DDOC, ASICE, PDF):

- The service is loaded under 1MB containers with two valid signatures
- The service is loaded around 5MB containers with two valid signatures
- The service is loaded near 10MB containers with two valid signatures
- The service is loaded under 1MB containers with ten valid signatures

The thread count will be increased from 5 to 60 with step of 5. The results are presented as throughput-latency graphs for each run.

15 References

16 References

16.1 Appendix 1 - Non-Functional Requirements

16.2 Appendix 2 - Functional Requirements

16.3 Appendix 3 - Validation Policy

17 Validation policy

!!! note Information will be added after analysis has been completed and agreed upon

17.1 Appendix 4 - Validation Constraint Configuration

17.2 Appendix 5 - Test Case Descriptions

18 List of Test Cases

18.1 BdocValidationFail.java

TestCaseID: Bdoc-ValidationFail-1

- TestType: Automated
- RequirementID:
- Title: Bdoc with single invalid signature
- Expected Result: The document should fail the validation
- File: IB-3960_bdoc2.1_TSA_SignatureValue_altered.bdoc

TestCaseID: Bdoc-ValidationFail-2

- TestType: Automated
- RequirementID:
- Title: Bdoc with multiple invalid signatures
- Expected Result: The document should fail the validation
- File: BdocMultipleSignaturesInvalid.bdoc

TestCaseID: Bdoc-ValidationFail-3

- TestType: Automated
- RequirementID:
- Title: Bdoc with multiple signatures both valid and invalid
- Expected Result: The document should fail the validation
- File: BdocMultipleSignaturesMixedWithValidAndInvalid.bdoc

TestCaseID: Bdoc-ValidationFail-4

- TestType: Automated
- RequirementID:
- Title: Bdoc with no signatures
- Expected Result: The document should fail the validation
- File: BdocContainerNoSignature.bdoc

TestCaseID: Bdoc-ValidationFail-5

- TestType: Automated
- RequirementID:
- Title: Bdoc with invalid mimetype in manifest
- Expected Result: document malformed error should be returned
- File: 23147_weak-warning-sha1-invalid-mimetype-in-manifest.bdoc

18.2 BdocValidationPass.java

TestCaseID: Bdoc-ValidationPass-1

- TestType: Automated
- RequirementID:
- Title: Bdoc with single valid signature
- Expected Result: The document should pass the validation
- File: Valid_ID_sig.bdoc

TestCaseID: Bdoc-ValidationPass-2

- TestType: Automated
- RequirementID:
- Title: Bdoc with multiple valid signatures
- Expected Result: The document should pass the validation

- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: Bdoc-ValidationPass-3

- TestType: Automated
- RequirementID:
- Title: Bdoc with warning on signature
- Expected Result: The document should pass the validation but warning should be returned
- File:

Attention! This test is disabled, check GitHub for specifics

18.3 DdocValidationFail.java

TestCaseID: Ddoc-ValidationFail-1

- TestType: Automated
- RequirementID:
- Title: Ddoc with single invalid signature
- Expected Result: The document should fail the validation
- File: test1-ddoc-revoked.ddoc

TestCaseID: Ddoc-ValidationFail-2

- TestType: Automated
- RequirementID:
- Title: Ddoc with multiple invalid signatures
- Expected Result: The document should fail the validation
- File: multipleInvalidSignatures.ddoc

TestCaseID: Ddoc-ValidationFail-3

- TestType: Automated
- RequirementID:
- Title: Ddoc with multiple signatures both valid and invalid
- Expected Result: The document should fail the validation
- File: multipleValidAndInvalidSignatures.ddoc

TestCaseID: Ddoc-ValidationFail-4

- TestType: Automated
- RequirementID:
- Title: Ddoc with no signatures
- Expected Result: The document should fail the validation
- File: DdocContainerNoSignature.bdoc

18.4 DdocValidationPass.java

TestCaseID: Ddoc-ValidationPass-1

- TestType: Automated
- RequirementID:
- Title: Ddoc with single valid signature
- Expected Result: The document should pass the validation
- File: 23734-ddoc13-13basn1.ddoc

TestCaseID: Ddoc-ValidationPass-2

- TestType: Automated
- RequirementID:
- Title: Ddoc v1.0 with multiple valid signatures
- Expected Result: The document should pass the validation
- File: DigiDoc_1.0_Tartu_ja_Tallinna_koostooleping.ddoc

Attention! This test is disabled, check GitHub for specifics

TestCaseID: Ddoc-ValidationPass-3

- TestType: Automated
- RequirementID:
- Title: Ddoc v1.1 with multiple valid signatures v1.1
- Expected Result: The document should pass the validation
- File: igasugust1.1.ddoc

TestCaseID: Ddoc-ValidationPass-4

- TestType: Automated
- RequirementID:
- Title: Ddoc v1.2 with multiple valid signatures
- Expected Result: The document should pass the validation
- File: igasugust1.2.ddoc

TestCaseID: Ddoc-ValidationPass-5

- TestType: Automated
- RequirementID:
- Title: Ddoc v1.3 with multiple valid signatures
- Expected Result: The document should pass the validation
- File: igasugust1.3.ddoc

18.5 DocumentFormatTests.java

TestCaseID: DocumentFormat-1

- TestType: Automated
- RequirementID: (TBD)
- Title: Validation of pdf document acceptance
- Expected Result: Pdf is accepted and correct signature validation is given
- File: hellopades-pades-lt-sha256-sign.pdf

TestCaseID: DocumentFormat-2

- TestType: Automated
- RequirementID: (TBD)
- Title: Validation of bdoc document acceptance
- Expected Result: Bdoc is accepted and correct signature validation is given
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: DocumentFormat-3

- TestType: Automated
- RequirementID: (TBD)
- Title: Validation of txt document rejection
- Expected Result: Txt document is rejected and proper error message is given
- File: hellopades-pades-lt-sha256-sign.txt

TestCaseID: DocumentFormat-4

- TestType: Automated
- RequirementID: (TBD)
- Title: Validation of p7s document rejection
- Expected Result: P7s document is rejected and proper error message is given
- File: hellocades.p7s

TestCaseID: DocumentFormat-5

- TestType: Automated
- RequirementID: (TBD)
- Title: Validation of zip document rejection
- Expected Result: Zip document is rejected and proper error message is given
- File: 42.zip

TestCaseID: DocumentFormat-6

- TestType: Automated
- RequirementID: (TBD)
- Title: Validation of doc document rejection
- Expected Result: Doc document is rejected and proper error message is given
- File: hellopades-pades-lt-sha256-sign.doc

TestCaseID: DocumentFormat-7

- TestType: Automated
- RequirementID: (TBD)
- Title: Validation of XML document rejection
- Expected Result: XML document is rejected and proper error message is given
- File: XML.xml

TestCaseID: DocumentFormat-8

- TestType: Automated
- RequirementID: (TBD)
- Title: Validation of png document rejection
- Expected Result: Png document is rejected and proper error message is given
- File: Picture.png

18.6 LargeFileTests.java

TestCaseID: PDF-LargeFiles-1

- TestType: Automated
- RequirementID:
- Title: Larger signed PDF files (PAdES Baseline LT).
- Expected Result: Bigger documents with valid signature should pass
- File: scout_x4-manual-signed_lt_9mb.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-LargeFiles-2

- TestType: Automated
- RequirementID:
- Title: Larger signed PDF files (PAdES Baseline LT).
- Expected Result: Bigger documents with valid signature should pass
- File: scout_x4-manual-signed_lta_9mb.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-LargeFiles-3

- TestType: Automated
- RequirementID:
- Title: Larger signed PDF files (PAdES Baseline LT).
- Expected Result: Bigger documents with valid signature should pass
- File: singlesignature_lt_1-2mb.pdf

TestCaseID: PDF-LargeFiles-4

- TestType: Automated
- RequirementID:
- Title: Larger signed PDF files (PAdES Baseline LT).
- Expected Result: Bigger documents with valid signature should pass
- File: digidocservice-signed-lta-1-2mb.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-LargeFiles-5

- TestType: Automated
- RequirementID:

- Title: Larger signed PDF files (PAdES Baseline LT).
- Expected Result: Bigger documents with valid signature should pass
- File: singlesignature_lt_3-8mb.pdf

TestCaseID: PDF-LargeFiles-6

- TestType: Automated
- RequirementID:
- Title: Larger signed PDF files (PAdES Baseline LT).
- Expected Result: Bigger documents with valid signature should pass
- File: egovenrment-benchmark-lta-3-8mb.pdf

Attention! This test is disabled, check GitHub for specifics

18.7 PdfBaselineProfileTests.java

TestCaseID: PDF-BaselineProfile-1

- TestType: Automated
- RequirementID:
- Title: The PDF has PAdES-B profile signature
- Expected Result: Document validation should fail
- File: hellopades-pades-b-sha256-auth.pdf

TestCaseID: PDF-BaselineProfile-2

- TestType: Automated
- RequirementID:
- Title: The PDF has PAdES-T profile signature
- Expected Result: Document validation should fail
- File:

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-BaselineProfile-3

- TestType: Automated
- RequirementID:
- Title: The PDF has PAdES-LT profile signature
- Expected Result: Document validation should pass
- File: hellopades-pades-lt-sha256-sign.pdf

TestCaseID: PDF-BaselineProfile-4

- TestType: Automated
- RequirementID:
- Title: The PDF has PAdES-LTA profile signature
- Expected Result: Document validation should pass
- File:

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-BaselineProfile-5

- TestType: Automated
- RequirementID:
- Title: The PDF has PAdES-LT and B profile signature
- Expected Result: Document validation should fail
- File: hellopades-lt-b.pdf

TestCaseID: PDF-BaselineProfile-6

- TestType: Automated
- RequirementID:
- Title: PDF document message digest attribute value does not match calculate value
- Expected Result: Document validation should fail
- File: hellopades-lt1-lt2-wrongDigestValue.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-BaselineProfile-7

- TestType: Automated
- RequirementID:
- Title: PDF file with a serial signature
- Expected Result: Document signed with multiple signers with serial signatures should pass
- File: hellopades-lt1-lt2-Serial.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-BaselineProfile-8

- TestType: Automated
- RequirementID:
- Title: PDF document signed with multiple signers parallel signature
- Expected Result: Document with parallel signatures should pass
- File: hellopades-lt1-lt2-parallel3.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-BaselineProfile-9

- TestType: Automated
- RequirementID:
- Title: PDF document signed with multiple signers parallel signature without Sscd
- Expected Result: Document with no qualified and without SSCD should fail
- File: hellopades-lt1-lt2-parallel3.pdf

18.8 PdfSignatureCryptographicAlgorithmTests.java

TestCaseID: PDF-SigCryptoAlg-1

- TestType: Automated
- RequirementID:
- Title: SHA512 algorithms (PAdES Baseline LT)
- Expected Result: Document signed with SHA512 algorithm should pass
- File: hellopades-lt-sha512.pdf

TestCaseID: PDF-SigCryptoAlg-2

- TestType: Automated
- RequirementID:
- Title: SHA1 algorithms (PAdES Baseline LT)
- Expected Result: Document signed with SHA1 algorithm should pass
- File: hellopades-lt-sha1.pdf

TestCaseID: PDF-SigCryptoAlg-3

- TestType: Automated
- RequirementID:
- Title: ECDSA algorithms (PAdES Baseline LT)
- Expected Result: Document signed with ECDSA algorithm should pass
- File: hellopades-ecdsa.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-SigCryptoAlg-4

- TestType: Automated
- RequirementID:
- Title: ECDSA224 algorithms (PAdES Baseline LT)
- Expected Result: Document signed with ECDSA224 algorithm should pass
- File: hellopades-lt-sha256-ec224.pdf

TestCaseID: PDF-SigCryptoAlg-5

- TestType: Automated
- RequirementID:
- Title: ECDSA256 algorithms (PAdES Baseline LT)
- Expected Result: Document signed with ECDSA256 algorithm should pass
- File: hellopades-lt-sha256-ec256.pdf

TestCaseID: PDF-SigCryptoAlg-6

- TestType: Automated
- RequirementID:
- Title: RSA1024 algorithms (PAdES Baseline LT)
- Expected Result: Document signed with RSA1024 algorithm should pass
- File: hellopades-lt-sha256-rsa1024.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-SigCryptoAlg-7

- TestType: Automated
- RequirementID:
- Title: RSA1023 algorithms (PAdES Baseline LT)
- Expected Result: Document signed with RSA1023 algorithm should pass
- File: hellopades-lt-sha256-rsa1023.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-SigCryptoAlg-8

- TestType: Automated
- RequirementID:
- Title: RSA2047 algorithms (PAdES Baseline LT)
- Expected Result: Document signed with RSA2047 algorithm should pass
- File: hellopades-lt-sha256-rsa2047.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-SigCryptoAlg-9

- TestType: Automated
- RequirementID:
- Title: RSA2048 algorithms (PAdES Baseline LT)
- Expected Result: Document signed with RSA2048 algorithm should pass
- File: PdfValidSingleSignature

18.9 PdfValidationFail.java

TestCaseID: PDF-ValidationFail-1

- TestType: Automated
- RequirementID:
- Title: The PDF-file has been signed with certificate that is expired before signing (PAdES Baseline LT)
- Expected Result: Document signed with certificate that expired before signing should fail.
- File: hellopades-lt-rsa1024-sha1-expired.pdf

TestCaseID: PDF-ValidationFail-2

- TestType: Automated
- RequirementID:
- Title: The PDF-file has been signed with expired certificate (PAdES Baseline LT)
- Expected Result: Document signed with certificate that is expired should fail.
- File: hellopades-lt-rsa1024-sha1-expired.pdf

TestCaseID: PDF-ValidationFail-3

- TestType: Automated
- RequirementID:
- Title: The PDF-file has been signed with revoked certificate (PAdES Baseline LT)

- Expected Result: Document signed with certificate that is revoked should fail.
- File: pades_lt_revoked.pdf

TestCaseID: PDF-ValidationFail-4

- TestType: Automated
- RequirementID:
- Title: The PDF-file has been signed with certificate that missing signed attribute (PAdES Baseline LT)
- Expected Result: PDF-file validation should fail
- File: missing_signing_certificate_attribute.pdf

TestCaseID: PDF-ValidationFail-5

- TestType: Automated
- RequirementID:
- Title: The PDF-file has been signed with certificate which has no non repudiation key usage attribute (PAdES Baseline LT)
- Expected Result: The PDF-file validation should fail with error.
- File: hellopades-pades-lt-sha256-auth.pdf

TestCaseID: PDF-ValidationFail-6

- TestType: Automated
- RequirementID:
- Title: hellopadess been signed with an expired certificate, where signing time is within the original validity
- Expected Result: Document signed with expired certificate should fail
- File: hellopades-lt-sha256-rsa2048-expired.pdf

TestCaseID: PDF-ValidationFail-7

- TestType: Automated
- RequirementID:
- Title: hellopadess been signed with an expired certificate, where signing time is within the original validity
- Expected Result: Document signed with expired certificate should fail
- File: hellopades-lt-sha256-rsa1024-expired2.pdf

TestCaseID: PDF-ValidationFail-8

- TestType: Automated
- RequirementID:
- Title: hellopadess been signed with an expired certificate, where signing time is within the original validity
- Expected Result: Document signed with expired certificate should fail
- File: hellopades-lt-sha1-rsa1024-expired2.pdf

18.10 PdfValidationPass.java

TestCaseID: PDF-ValidationPass-1

- TestType: Automated
- RequirementID:
- Title: The PDF-file has been signed with certificate that is expired after signing (PAdES Baseline LT)
- Expected Result: Document signed with certificate that expired after signing should pass.
- File: hellopades-lt-sha256-rsa1024-not-expired.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-ValidationPass-2

- TestType: Automated
- RequirementID:
- Title: The PDF-file has been signed with certificate that will expire in 7 days after signing (PAdES Baseline LT)
- Expected Result: Document signed with certificate that expired after signing should pass.
- File: hellopades-lt-sha256-rsa2048-7d.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-ValidationPass-3

- TestType: Automated
- RequirementID:
- Title: Certificate contents are include in response (PAdES Baseline LT)
- Expected Result: The PDF-file validation should pass
- File: hellopades-lt-sha256-ocsp-15min1s.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-ValidationPass-4

- TestType: Automated
- RequirementID:
- Title: Pdf with single valid signature
- Expected Result: Document should pass.
- File: PdfValidSingleSignature.pdf

18.11 SignatureRevocationValueTests.java

TestCaseID: PDF-SigRevocVal-1

- TestType: Automated
- RequirementID:
- Title: The PDF-file has PAdES-LT profile signature and an OCSP confirmation that is more than 15 minutes later than the signatures Time Stamp.
- Expected Result: Document with ocsp over 15 min delay should pass but warn
- File: hellopades-lt-sha256-ocsp-15min1s.pdf

TestCaseID: PDF-SigRevocVal-2

- TestType: Automated
- RequirementID:
- Title: The PDF-file has PAdES-LT profile signature and an OCSP confirmation that is more than 15 minutes later than the signatures Time Stamp.
- Expected Result: Document with ocsp over 15 min delay should pass but warn
- File: hellopades-lt-sha256-ocsp-15min1s.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-SigRevocVal-3

- TestType: Automated
- RequirementID:
- Title: The PDF-file has PAdES-LT profile signature and an OCSP confirmation more than 24 hours later than the signatures Time Stamp.
- Expected Result: Document with over 24h delay should fail
- File: hellopades-lt-sha256-ocsp-28h.pdf

TestCaseID: PDF-SigRevocVal-4

- TestType: Automated
- RequirementID:
- Title: The PDF-file has PAdES-LT profile signature and an OCSP confirmation more than 24 hours later than the signatures Time Stamp.
- Expected Result: Document with over 24h delay should fail
- File: hellopades-lt-sha256-ocsp-28h.pdf

TestCaseID: PDF-SigRevocVal-5

- TestType: Automated
- RequirementID:
- Title: The PDF-file has been signed with PAdES Baseline LTA profile signature, the signature contains CRL.
- Expected Result: Document with no ocsp or crl in signature should fail
- File: hellopades-lta-no-ocsp.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: PDF-SigRevocVal-6

- TestType: Automated
- RequirementID:
- Title: PDF signature has OCSP confirmation before Time Stamp
- Expected Result: Document signed with ocsp time value before best signature time should fail
- File: hellopades-lt-sha256-rsa2048-ocsp-before-ts.pdf

Attention! This test is disabled, check GitHub for specifics

18.12 ValidationReportJsonStructureVerification.java

TestCaseID: Bdoc-ValidationReport-1

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (Bdoc valid single signature)
- Expected Result: All required elements are present according to BdocDocSimpleReportSchema.json
- File: Valid_ID_sig.bdoc

TestCaseID: Bdoc-ValidationReport-2

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (Bdoc valid multiple signatures)
- Expected Result: All required elements are present according to SimpleReportSchema.json
- File: Baltic MoU digital signing_EST_LT_LV.bdoc

Attention! This test is disabled, check GitHub for specifics

TestCaseID: Bdoc-ValidationReport-3

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (Bdoc invalid single signature)
- Expected Result: All required elements are present according to BdocDocSimpleReportSchema.json
- File: IB-3960_bdoc2.1_TSA_SignatureValue_altered.bdoc

TestCaseID: Bdoc-ValidationReport-4

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (Bdoc indeterminate status)
- Expected Result: All required elements are present according to BdocDocSimpleReportSchema.json
- File: test1-bdoc-unknown.bdoc

TestCaseID: Bdoc-ValidationReport-5

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Check for optional subindication and error elements
- Expected Result: Error and subindication elements are present
- File: IB-3960_bdoc2.1_TSA_SignatureValue_altered.bdoc

TestCaseID: Bdoc-ValidationReport-6

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Check for optional warning element
- Expected Result: Warning element is present
- File: 23154_test1-old-sig-sigat-NOK-prodat-OK-1.bdoc

TestCaseID: Bdoc-ValidationReport-7

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Bdoc report with no signatures
- Expected Result: Report is returned with required elements
- File:BdocContainerNoSignature.bdoc

TestCaseID: Pdf-ValidationReport-8

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (Pdf valid single signature)
- Expected Result: All required elements are present according to SimpleReportSchema.json
- File: hellopades-lt-sha256-ec256.pdf

TestCaseID: Pdf-ValidationReport-9

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (Pdf valid Multiple signatures)
- Expected Result: All required elements are present according to SimpleReportSchema.json
- File:

TestCaseID: Pdf-ValidationReport-10

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (Pdf invalid signature)
- Expected Result: All required elements are present according to SimpleReportSchema.json
- File: hellopades-lt-b.pdf

Attention! This test is disabled, check GitHub for specifics

TestCaseID: Pdf-ValidationReport-11

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (Pdf indeterminate status)
- Expected Result: All required elements are present according to SimpleReportSchema.json
- File: hellopades-lt-rsa1024-sha1-expired.pdf

TestCaseID: Pdf-ValidationReport-12

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Pdf report with no signatures
- Expected Result: Report is returned with required elements
- File: PdfNoSignature.pdf

TestCaseID: Ddoc-ValidationReport-13

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (ddoc valid single signature)
- Expected Result: All required elements are present according to SimpleReportSchema.json
- File: 18912.ddoc

TestCaseID: Ddoc-ValidationReport-14

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (ddoc valid Multiple signatures)
- Expected Result: All required elements are present according to SimpleReportSchema.json
- File: igasugust1.1.ddoc

TestCaseID: Ddoc-ValidationReport-15

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (ddoc invalid signature)
- Expected Result: All required elements are present according to SimpleReportSchema.json
- File: test1-ddoc-revoked.ddoc

TestCaseID: Ddoc-ValidationReport-16

- TestType: Automated

- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: JSON structure has all elements (ddoc indeterminate status)
- Expected Result: All required elements are present according to SimpleReportSchema.json
- File: test1-ddoc-unknown.ddoc

TestCaseID: Ddoc-ValidationReport-17

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Check for optional subindication and error elements
- Expected Result: Error and subindication elements are present
- File: test1-ddoc-unknown.ddoc

TestCaseID: Ddoc-ValidationReport-18

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Check for optional warning element
- Expected Result: Warning element is present
- File:

Attention! This test is disabled, check GitHub for specifics

TestCaseID: Ddoc-ValidationReport-19

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Ddoc report with no signatures
- Expected Result: Report is returned with required elements
- File: DdocContainerNoSignature.ddoc

18.13 ValidationReportValueVerification.java

TestCaseID: Bdoc-ValidationReportValue-1

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of values in Validation Report XAdES_BASELINE_LT_TM, QES, FullSignatureScope
- Expected Result: All required elements are present and meet the expected values.
- File: Valid_ID_sig.bdoc

TestCaseID: Bdoc-ValidationReportValue-2

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of values in Validation Report XAdES_BASELINE_LT, QES, FullSignatureScope
- Expected Result: All required elements are present and meet the expected values.
- File: 23635_bdoc_ts_OCSP_random_nonce.bdoc

TestCaseID: Bdoc-ValidationReportValue-3

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of values in Validation Report XAdES_BASELINE_LT, AdES, FullSignatureScope
- Expected Result: All required elements are present and meet the expected values.
- File: 23154_test1-old-sig-sigat-NOK-prodat-OK-1.bdoc

TestCaseID: Bdoc-ValidationReportValue-4

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of values in Validation Report XAdES_BASELINE_LT_TM, AdESqc, FullSignatureScope
- Expected Result: All required elements are present and meet the expected values.
- File: 23200_weakdigest-wrong-nonce.asice

Attention! This test is disabled, check GitHub for specifics

TestCaseID: Bdoc-ValidationReportValue-5

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of values in Validation Report XAdES_BASELINE_LTA, QES, FullSignatureScope
- Expected Result: All required elements are present and meet the expected values.
- File: EE_SER-AEX-B-LTA-V-24.pdf

TestCaseID: Ddoc-ValidationReportValue-6

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of values in Validation Report, xml v1.0, checks for missing info
- Expected Result: All required elements are present and meet the expected values and other values are empty as expected.
- File: ICT_MoU_FI-EE_10dec2013OneSignature.bdoc

Attention! This test is disabled, check GitHub for specifics

TestCaseID: Ddoc-ValidationReportValue-7

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of values in Validation Report, xml v1.1, checks for missing info
- Expected Result: All required elements are present and meet the expected values and other values are empty as expected.
- File: Igasugust1.1.ddoc

TestCaseID: Ddoc-ValidationReportValue-8

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of values in Validation Report, xml v1.2, checks for missing info
- Expected Result: All required elements are present and meet the expected values and other values are empty as expected.
- File: Igasugust1.2.ddoc

TestCaseID: Ddoc-ValidationReportValue-9

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of values in Validation Report, xml v1.3, checks for missing info
- Expected Result: All required elements are present and meet the expected values and other values are empty as expected.
- File: Igasugust1.3.ddoc

18.14 ValidationRequestTests.java

TestCaseID: ValidationRequest-1

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Input random base64 string as document with bdoc document type
- Expected Result: Error is returned stating problem in document
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-2

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Input random base64 string as document with pdf document type
- Expected Result: Error is returned stating problem in document
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-3

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Input random base64 string as document with ddoc document type
- Expected Result: Error is returned stating problem in document
- File:

TestCaseID: ValidationRequest-4

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Input a request with empty values
- Expected Result: Errors are returned stating the missing values
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-5

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Request with not base64 string as document
- Expected Result: Error is returned stating encoding problem
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-6

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of wrong report type as input
- Expected Result: Error is returned stating wrong report type
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-7

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of valid (but not simple) report type as input
- Expected Result: Correct report is returned (currently simple report is returned in all cases)
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-8

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of wrong document type as input
- Expected Result: Correct error code is returned
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-9

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Mismatch in documentType and actual document (bdoc and pdf)
- Expected Result: Error is returned
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-10

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Mismatch in documentType and actual document (asice and bdoc)
- Expected Result: Error is returned
- File: TS-11_23634_TS_2_timestamps.asice

Attention! This test is disabled, check GitHub for specifics

TestCaseID: ValidationRequest-11

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/

- Title: Verification of case insensitivity in document type
- Expected Result: Report is returned
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-12

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Verification of filename value (filename do not match the actual file)
- Expected Result: The same filename is returned as sent in the request
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-13

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Request has invalid character in filename
- Expected Result: Correct error code is returned
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-14

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Request has invalid key on document position
- Expected Result: Error is returned
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-15

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Request has wrong key on report type position
- Expected Result: Error is returned
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-16

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Request has XML as document type (special case, XML is similar to ddoc and was a accepted document type in earlier versions)
- Expected Result: Error is given
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-17

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Request has long (38784 characters) in filename field
- Expected Result: Report is returned with the same filename
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-18

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Totally empty request body is sent
- Expected Result: Error is given
- File: None

TestCaseID: ValidationRequest-19

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Request with more parameters than expected is sent
- Expected Result: Error is given or extra parameters are ignored?
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-20

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Request with special chars is sent
- Expected Result: Validation report is returned
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-21

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Mismatch in documentType and actual document (ddoc and bdoc)
- Expected Result: Error is returned
- File: Valid_IDCard_MobID_signatures.bdoc

TestCaseID: ValidationRequest-22

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Mismatch in documentType and actual document (ddoc and pdf)
- Expected Result: Error is returned
- File: PdfValidSingleSignature.pdf

TestCaseID: ValidationRequest-23

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Mismatch in documentType and actual document (bdoc and pdf)
- Expected Result: Error is returned
- File: PdfValidSingleSignature.pdf

TestCaseID: ValidationRequest-24

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Mismatch in documentType and actual document (bdoc and ddoc)
- Expected Result: Error is returned
- File: igasugust1.3.ddoc

TestCaseID: ValidationRequest-25

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Mismatch in documentType and actual document (pdf and ddoc)
- Expected Result: Error is returned
- File: igasugust1.3.ddoc

TestCaseID: ValidationRequest-26

- TestType: Automated
- RequirementID: http://open-eid.github.io/SiVa/siva/interface_description/
- Title: Mismatch in documentType and actual document (pdf and bdoc)
- Expected Result: Error is returned
- File: Valid_IDCard_MobID_signatures.bdoc