Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Federal Department of Finance FDF

Federal Office of Information Technology, Systems and Telecommunication FOITT

FOITT, November 2025

Discreet Validator Service 3.0

Interface Specification (Version 1.3.0)

Table of contents

1 Introduction

This document describes the interface to the so-called discrete validator. The discrete validator (hereafter DV) is a REST service that can check signatures on PDF documents without receiving the PDF documents. The caller of the web service only transmits the signatures on the document and the hash of the document.

For the discrete validator, the "Java Client" so named in the contract between the cantons and eOperations Schweiz is delivered with an SDK (Software Development Kit) and a CLI (command line interface) client (hereafter "Java Client Software Package"). The SDK provides an easy-to-use abstraction layer for the validation of documents. It takes care of extracting the data required for calling the web service from the PDF documents to be validated and executes the call to the web service.

2 Overview and validation process

The discrete validation of a PDF document is based on the verification of signatures by means of certificates stored in the database.

Rough procedure of the validation:

1. User starts local client
2. select the document to be validated
3. select client (from master data list)
4. select signature
5. send
6. signatures are extracted and hashes are calculated (locally)
7. data is transferred to the server
8. server checks signature
9. hash in the signature is decrypted with the public key and compared with the locally calculated one
10. check certificate chain
11. check client
12. check CRL (certificate revocation list) and TSA (timestamp authority)
13. create and send back the verification report (optional)
14. client displays result
15. client can save PDF report

2.1 Validation reference

The web validator can be used to check the validation results:

Productive-System:

- https://www.validator.ch
- https://www.egovsigval.bit.admin.ch/

Test-System (acceptance):

- https://www.egovsigval-a.bit.admin.ch/

2.1.1 Monitoring Endpoint

Backend URL's:
URL (production): https://egovsigval-backend.bit.admin.ch/public/api/document-type
URL (acceptance): https://egovsigval-backend-a.bit.admin.ch/public/api/document-type

2.2 Use of the discrete validator

To use the DV, the following requirements must be met:

– A username and password are required to use the DV. Customers who have already used the previous (old) DV in the past and therefore already have a username and password, need a new username and password. These are created by BIT (OFIT / UFIT / FOITT) for all environments (REF / ABN / PROD) and sent to the customers.

    - Cantons receive the username and password directly from the BIT after signing the corresponding contract with eOperations Schweiz. Municipalities receive the username and password of their canton from the responsible office of the canton.
    - Customers within the federal administration should consult the BIT website (Signatur-Verifikationsdienst (SD) (admin.ch))

The Java Client software package can be requested as follows:

- Cantons: Via eOperations Schweiz
- Municipalities: Via canton
- Customers within the federal administration: These customers please consult the BIT offer page (Signatur-Verifikationsdienst (SD) (admin.ch)).

Basically, the DV works as before. The information on how to carry out a validation can be found under docs/index.html in the Java client software package supplied.

The latest Java client software package is made available for cantons via eOperations Schweiz and for customers within the federal administration by BIT.

3 CLI (command line client)

Command line client (CLI) can be used from a windows (PowerShell) or Linux console (e.g. Bash). Manual of the CLI is part of the distribution package. The manual can be found in *docs* folder and named *index.html*.

The CLI package can be placed in any local system folder. It can be started directly from the installation folder. Unzip the package, go into the home folder of the package and start the cli as depicted below.

The CLI requires a Java Runtime Environment of version 1.8 or above. The Java executable must be referenced via environment variable JAVA_HOME.

From JDK version 9 and higher, the default keystore type is PKCS12.
So, if you are running the CLI Validator on a higher version than Java 8 you need to set this properties in the validate.bat / validate.sh script.

```
"DEFAULT_JVM_OPTS= -Djavax.net.ssl.trustStore=truststore.jks -
Djavax.net.ssl.trustStorePassword=changeit -
Djavax.net.ssl.trustStoreType=jks"
```

The overview of the actual possible mandators you will find in chapter 8.

Executable scripts for PowerShell and Linux/Bash.

- validate.bat (PowerShell)
- validate.sh (Linux/Bash)

usage: CommandLineInterfaceClient

Command	Detail	Description
-b	--batch-input-dir <dir>	Path to the directory containing the files to be validated. The option -b is mutually exclusive with -f.
-c	--container-check	Container check validates all signatures in the pdf file. The option -c is mutually exclusive with -s.
-d	--dump	Logs the JSON object of the request and response.
-e	--unsigned	Generate report even for unsigned files
-E	--cert	File name and optional password to use for client certificate authentication. Supported file types are PKCS12 and JKS.
-f	--pdffile <filename>	file to validate
-i	--config <filename>	Name of config file to use
-l	--lang <lang>	get pdf report in the given language, supported codes: de, fr, it, en. This is an optional parameter, if omitted de is used.
-list		List digital signatures of given PDF file
-m	--mandator <mandator >	mandator to use, e.g. Qualified, FullQualified, Strafregisterauszug (Can also be defined in config file)
-o	--report <filename> --report <dir>	When -f is used, a PDF report will be saved at the given name. When -b is used, a batch report and a report for each file will be generated and packaged into a ZIP file. The ZIP will be saved at the given name. The batch report will be named "batch_report.pdf" The report for each file will be named after the validated document + "_report.pdf" Example: -> C:\tmp\example_report.zip -> C:\tmp\example_report.zip (contains batch_report.pdf and per-file reports)
-p	--proxy <proxyhost:proxyport:proxyuser:proxypass>	HTTP Proxy host and port to use for accessing the validation webservice. If not present, the Java System Properties are checked. For authentication use the format proxy:port:user:password
-pw	--password <password>	password for the basic authentication
-rs	--outputstyle <style>	Defines the style of the generated pdf report, either a

Command	Detail	Description
		container with a report per signature or one report for all.The valid options are: -rs single or -rs container. The option can be omitted; default is single mode.
-s	--signature <signaturename>	Name of a signature to check in this pdf file (e.g. Signature1). Works only for not nested documents. The option -s is mutually exclusive with -c.
-sun	--serviceusername	Optional parameter for examiner username and can be passed to the validation service. As a result, these values are included in the resulting validation reports.
-suo	--serviceuserorganization	Optional parameter for examiner organization can be passed to the validation service. As a result, these values are included in the resulting validation reports.
-u	--url <url>	URL of the validation webservice. (Can also be defined in config file)
-un	--username <username>	username for the basic authentication

3.1 Example of CLI call

Validation of all signatures in a file

```
validate -u https://egovsigval-backend.bit.admin.ch -m Kanton-Zug-
Finanzdirektion -f TC072_Zug_Beschwerdeschrift-Nr2-conv-sig-
Eingangsstempel.pdf -c
```

Validation of specific signature

```
validate -u https://egovsigval-backend.bit.admin.ch -m myvalidation -f
file.pdf -s signature1
```

Generate validation report

```
validate -u https://egovsigval-backend.bit.admin.ch -m myvalidation -f
file.pdf -c -o report.pdf
```

Creating a validation report for multiple files (batch)

This example assumes that

- the name of the directory containing the pdf files to be validated is C:\Users\validator\input,
- that you want to apply the validation rules for myvalidation.
- and that the validation service is running at https://egovsigval-backend.bit.admin.ch

```
validate -u https://egovsigval-backend.bit.admin.ch -m myvalidation -b
C:\Users\validator\inputdirectory -o batch_report.zip
```

3.2 Configuration File

Options for mandator, service URL and proxy settings can be configured in configuration file. Format of configuration file must be similar to a Java property file. Format of a configuration file:

```
mandator=<value>
validator_url=<value>
proxy_host=<value>
proxy_port=<value>
```

3.3 Error Messages and Logging

Info and error lgos can be configured in configuration file logback.xml, it is part of the Classes folder. This client is using Logback as framework of SLF4J. Details how the logging can be adjusted can be found on website: http://logback.qos.ch/manual/configuration.html#syntax

3.4 Client library

Client library offers a simple abstraction layer for the validation of a document. It offers all the functions needed to extract all information needed to use the validation service. It also offers a method to send the information to be validated to the discrete validation service. As a reply, the validation result including report is sent back.
The CLI client uses the client library and can be seen as an example of how the client library can be used.
Authentication credentials can be set in Class *ValidationServiceClient* by using method *credentials*.

```
ValidationServiceClient serviceClient =
ValidationServiceClientBuilder.newBuilder() //
                .serviceUrl(serviceUrl) //
                .credential(new UserPasswordCredential(username,
Secret.hide("password".toCharArray()))) //
                .build();
```

Java documentation of SDK is available in *docs/Javadoc* or as short into in *index.html*.

3.4.1 Java-Interface

Example: Validation of signature

- Variable pdfFile, file object pointing to PDF file
- Variable client describes mandate which should be used as reference for the validation rules
- Variable serviceURL defines end point of validation service

```
FileRequest fileRequest = new FileRequest(pdfFile, client);
ValidationServiceClient serviceClient =
ValidationServiceClientBuilder.newBuilder() //
                .serviceUrl(serviceUrl) //
                .build();
```

```
ValidationResponse response =
serviceClient.validateOneRequest(Arrays.asList(fileRequest), false,
null, null, "de", null);
```

Example: Validation of referenced signature

- Variable pdfBytes defines PDF file as byte[] object
- Variable sigName contains a string that specifies name of signature filed to be validated
- Variable client describes mandate which should be used as reference for the validation rules
- Variable serviceURL defines end point of validation service

```
ValidationServiceClient serviceClient =
ValidationServiceClientBuilder.newBuilder() //
                .serviceUrl(serviceUrl) //
                .build();
ValidationResponse response =
serviceClient.validateOneSignature(pdfBytes, client, false, sigName,
                "file.pdf", null, null, "de", null);
```

Example: Generation of validation report

- Variable pdfFile, file object pointing to PDF file
- Variable client describes mandate which should be used as reference for the validation rules
- Variable serviceURL defines end point of validation service

```
FileRequest fileRequest = new FileRequest(pdfFile, client);
ValidationServiceClient serviceClient =
ValidationServiceClientBuilder.newBuilder() //
                .serviceUrl(serviceUrl) //
                .build();
ValidationResponse response =
serviceClient.validateOneRequest(Arrays.asList(fileRequest), true,
null, null, "de", "report.pdf");
File file = new File(response.getPdfOutputFileName());
try (OutputStream stream = new FileOutputStream(file)) {
        stream.write(response.getPdfReport());
}
```

Example: Validating all signatures in a batch of files

- the variables pdfFile1 and pdfFile2 contain File objects pointing to PDF files on disk,
- that the variable client contains a String specifying the "client" property to use to validate the file,
- and that the variable serviceUrl contains a String specifying the base URL of the validation service server.

```
FileRequest fileRequest1 = new FileRequest(pdfFile1, client);
FileRequest fileRequest2 = new FileRequest(pdfFile2, client);
ValidationServiceClient serviceClient =
ValidationServiceClientBuilder.newBuilder() //
                .serviceUrl(serviceUrl) //
                .build();
```

```
ValidationResponse response =
serviceClient.validateOneRequest(Arrays.asList(fileRequest1,
fileRequest2), false, null, null, "de", null);
```

4 API Description

Discrete validation service can be used with RESTful API. The following return codes are defined:

Status Code	Detail
200	Request could be successfully processed. Response contains data that must be analyzed further. Response could contain error messages.
400	Bad Request in case request is malformed or contains wrong encoding.
404	Not Found – In case client cannot be found (mandant)
500	Internal Server Error. In case there is an error on the server side.

As response a JSON document is sent back. Content type of response is application/json. Character set is UTF-8 encoding.

Basic authentication is used to work directly with API. Username and password must be sent as part of the header.

4.1 Request Attributes

Name	Description	Type / Example	Required
language	Language of report	String / „de“, „fr“, „it“, „en“	yes
pdfOutputFileName	Name of output report, path to report can be defined. This information is used only locally.	String / C:\\Workspace\\ Qualified_TCI009_Test_ZertES_doppelt_Gears.report.pdf	no
pdfReport	Attribute to request PDF report	Boolean / true, false	yes
processUnsignedFiles	Attribute to request PDF report instead of error message	Boolean / true, false	yes
userName	Username or account name used for the validation	String	no
userOrganization	Name of organization to be used in validation process	String	No
validatableFiles	Object containing all information about file to be validated	JSON object (Array of documents)	No
documentHash	Hash of document to be validated	String / base64 encoded	Yes

Name	Description	Type / Example	Required
documentName	Name of document	String	Yes
Signatures	Object containing signatures of document	JSON object (Array of signatures)	Yes
client	Mandant of document to be used for validation	String	No
signatureContent	CMS object, signature data with certificate chain, signer certificate, timestamp,	Byte[] (String Base64-encoded, ASN.1)	Yes
signatureDate	Date of signature	https://www.rfc-editor.org/rfc/rfc3339#section-5.6	No
signatureDigest	Signed hash value	Byte[] (String Base64-encoded)	Yes
signatureName	Name of signature	String	No
signaturePosition	Position of signature (one or multiple signatures)	Integer / 0, 1, 2	Yes
changeLevel	Level of changes to be considered in validation	String / "permitted:sign", "ignorable"	Yes
coveringWholeDocument	Coverage of signature	Boolean / true, false	Yes
additionalInfos	Optional information	JSON object	No
reason	Reason of signature	String	No
validationData	JSON object containing certificates, crl or ocsp	JSON object	No
certificates	Certificate chain base64 encoded	JSON Array of Strings	No
crl	Certificate revocation list base 64 encoded	JSON Array of Strings	No
ocsp	OCSP base 64 encoded	JSON Array of Strings	No
client	Mandant to be used in validation of documents	String	Yes

4.1.1 Example of request

```
{
    "language": "de",
    "pdfOutputFileName": "C:\\Workspace\\ES_doppelt_Gears.report.pdf",
    "pdfReport": true,
    "processUnsignedFiles": true,
    "userName": null,
    "userOrganization": null,
    "validatableFiles": [{
               "documentHash": "JQXZkHMluP6hKDdjHcg…AAkqcrM84=",
               "documentName": "TCI009_Test_ZertES_doppelt_Gears.pdf",
               "signatures": [{
                       "client": null,
                       "signatureContent": "MIIX6AYJ…==",
                       "signatureDate": "2021-12-16T09:24:16Z",
                       "signatureDigest": "Lw5+…==",
                       "signatureName": "Signature1",
                       "signaturePosition": 0,
                       "changeLevel": "permitted:sign",
                       "coveringWholeDocument": false,
                       "additionalInfos": {
                               "reason": "Reason of signature 1"
                       }, {
                       "client": null,
                       "signatureContent": "MIIX5wY…XX==",
                       "signatureDate": "2021-12-16T09:26:37Z",
                       "signatureDigest": "eJ7sll…==",
                       "signatureName": "Signature2",
                       "signaturePosition": 1,
                       "changeLevel": "ignorable",
                       "coveringWholeDocument": false,
                       "additionalInfos": {
                               "reason": "Reason of signature 2"
                       }
               }],
               "validationData": {
                       "certificates": [],
                       "crl": [],
                       "ocsp": ["SDADFAS..FD+"]
               },
               "client": "Qualified"
       }]
}
```

4.2 Response Attributes

Name	Description	Type / Example
pdfReport	Report as base64 encoded PDF	String / base64 encoded
pdfOutputFileName	File name of report	String
fileReports	Report for validated files	JSON object (Array of documents)
signatureReports	Report for each signature of document	JSON object (Array of reports per signature)
signatureName	Name of signature validated	String
certificateDetails	Certificate details used for this signature	JSON object (Certificate details)
qualification	List of possible certificates	String: (FORTGESCHRITTEN | QUALIFIZIERT | ELDIV | GEREGELT)
classification	Certificate class	String: (CLASS_A | CLASS_B | CLASS_C | UNKNOWN)
type	Type of certificate in certificate chain	String: (ROOT | INTERMEDIATE | PERSON | ORGANIZATION |

Name	Description	Type / Example
		MACHINE | TIMESTAMP | UNKNOWN)
hardware	Private key generated on hardware	Boolean / true, false
approved		Boolean / true, false
swiss	Certificate of Swiss TSP	Boolean / true, false
mandatorDetails		
mandant	Mandant of document to be used for validation	String / Definition of mandants
description	Description of mandants	
link	Link to policies of mandant	
direct		Boolean / true, false
revocationDetails	Revocation Details	
state	Status revocation details	String: (REVOKED | NOT_REVOKED | UNKNOWN)
date	Date of revocation	Revocation date Example: "2021-12-16T09:26:46.012Z"
timestampDetails	Time stamp details	
status	Status of time stamp	String: (VALID | NOT_VALID | NOT_TRUSTED | UNKNOWN | MISSING)
subject	Time stamp description	String
signatureDate	Signature date	Date, example: "2021-12-16T09:26:46.012Z"
reports		
valid	Result of report	Status: (INFO | VALID | UNSURE | INVALID)
message	Additional info	
type	Type of validation	String: (INTEGRITY | CERTIFICATE | REVOCATION | MANDATOR | TIMESTAMP)
documentName	Name of report	String
documentHash	Document hash	Document hash
mandatorRequirements		
mandator	Name of mandator	
status	Status	String: (INFO | VALID | UNSURE | INVALID)

To explain: as a result, we get a list called "fileReports" in the top-level element. This list contains a list of all signatures for each file and in it a statement about the certificate applicable in the validator, the client and the revocation plus a statement about the validation status of the individual categories (integrity, certificate, revocation and client). The values of the states can assume VALID, INVALID and UNSURE.

- INTEGRITY: Document was not changed after signing.
- CERTIFICATE: The certificate was valid at the time of signing (i.e. not expired). The certificate is known to us, so it is trustworthy.
- REVOCATION: revocation check (invalidity, blocking of the certificate).
- If the signature is provided with a time stamp, the revocation is checked for this time. If no time stamp is included, the test date is relevant.
- TIMESTAMP: Time stamp check (if activated in the tenant). Verification that the time stamp is valid and applied by a trusted time stamp service.
- MANDATOR: Checks whether the signature certificate used is permissible for this client
- AdditionalInfo/Reason: Reason is only rendered in the PDF-Report (not in JSON Response).

Certificate Details:

- qualification: Qualification (QUALIFIED, REGULATED or ADVANCED)
- type: root (ROOT), intermediate root (INTERMEDIATE), person (PERSON),
- Organization (ORGANIZATION), machine (MACHINE), time stamp (TIMESTAMP),
- unknown (UNKNOWN)
- classification: bit class A (CLASS_A), B (CLASS_B), C (CLASS_C) or unknown (UNKNOWN)
- hardware: token (boolean; hardware certificate or software certificate)
- approved: Approved provider (boolean)
- swiss: Swiss provider (boolean)

4.2.1 Example of response

```
{
  "pdfReport": "JVBERi0xLjQKJfbk/N...9GCg==",
  "pdfOutputFileName": "Qualified_TCI009_report_with_change_doc.pdf",
  "fileReports": [
    {
      "signatureReports": [
        {
          "signatureName": "",
          "certificateDetails": {
            "qualification": "QUALIFIZIERT",
            "classification": "CLASS_B",
            "type": "UNKNOWN",
            "hardware": true,
            "approved": true,
            "swiss": false
          },
          "mandatorDetails": {
            "mandant": "Kanton-Zug-Finanzdirektion",
            "description": "",
            "link": "https://bit.admin.ch/sigval/policies",
            "direct": true
          },
          "revocationDetails": { "state": "NOT_REVOKED" },
          "timestampDetails": {
            "status": "VALID",
            "subject": "CN=Swiss Government TSA, OU=Time Stamp
Services, OU=Swiss Government PKI, O=Bundesamt fuer Informatik und
Telekommunikation (BIT), OID.2.5.4.97=VATCH-CHE-221.032.573, L=Bern,
C=CH",
            "signatureDate": "2021-12-16T09:26:46.012Z"
          },
          "reports": [
            { "valid": "VALID", "message": "CERTIFICATE", "type":
"CERTIFICATE" },
            { "valid": "VALID", "message": "INTEGRITY", "type":
"INTEGRITY" },
            { "valid": "INVALID", "message": "MANDATOR", "type":
"MANDATOR" },
            { "valid": "VALID", "message": "REVOCATION", "type":
"REVOCATION" },
            { "valid": "VALID", "message": "TIMESTAMP", "type":
"TIMESTAMP" }
          ]
        }
      ],
      "documentName": "TCI009_Test_ZertES_doppelt_Gears.pdf",
      "documentHash": "JQXZkHMluP6hKDdjHcglNqyNLyn8jInsEcAAkqcrM84=",
      "mandatorRequirements": {
        "mandator": "Kanton-Zug-Finanzdirektion",
        "status": "INVALID"
      }
    }
  ]
}
```

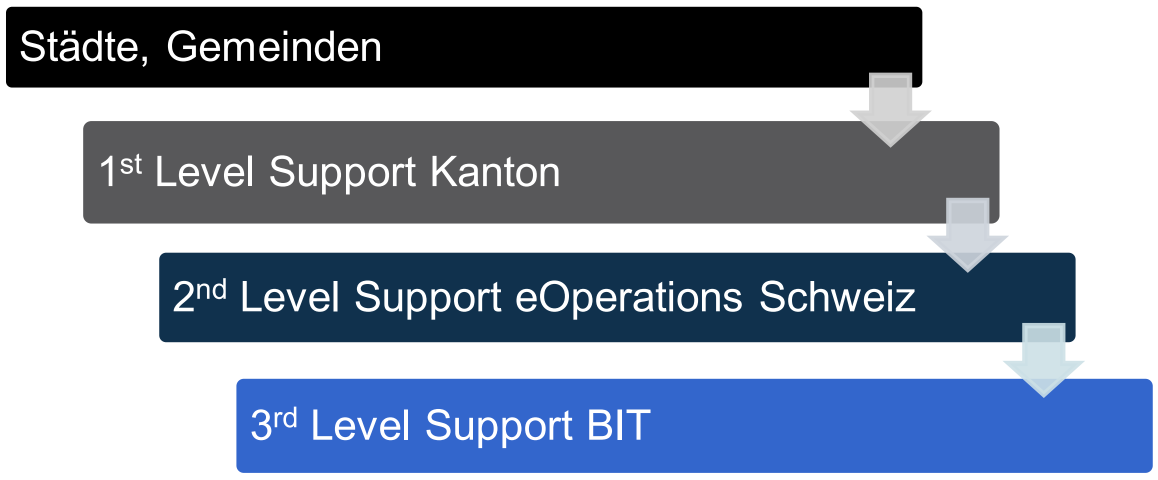
5 Possible Issues

Various errors can occur on different application levels:

1. Error with HTTP requests
2. Errors with web service; especially issues with service attributes that are required to apply the validation

5.1 Support

The following support process applies to the Discrete Validator:



Please contact the next point in the support cascade to submit your request. In projects, the support organization may differ from the above representation.

6 Example of document validation

6.1 Introduction

To be able to generate an appropriate validation report, the validation service requires besides the signature information also information about the document itself. Therefore, the validation interface has been extended in specification 1.2 (client 2.2.0) to be able to apply more exact valuation rules. Validation result is provided as JSON object.

Examples below show possible use cases:

6.2 Signature according to EÖBV

A valid PDF file according to EÖBV1 contains a qualified signature and a signature at the last position to specify the confirmation of function from registry of deed (UPReg).

Definition:

A deed must be a PDF/A-1 or PDF/A-2 conformance file, containing N+2 (N must be bigger or equal to 0) signatures in the following order:

1. First N signatures are qualified signatures (signatures of contract parties); these

   signatures are optional.
2. Qualified signature of notary
3. Regulated signature of registry of deed; signature is inserted in field named *RegSig.*
4. Regulated signature of registry of canton (this signature is optional and dependent on cantonal law)
5. Additional signatures (e.g. archiving solution) are not part of deed

A validation with CLI can be executed according to the following example:

```
validate.bat -c -u https://localhost:8080/validator/rs -m upreg-fn -f
examples\qualified-with-eobv.pdf -d –e
```

Following errors can occur:
- Function name is missing (only qualified signature available)
- Document has been amended after applying function confirmation
- Last signature is no function name confirmation (e.g. additional qualified signature at the end of confirmation of function)
- Validation of unsigned document

6.3 Function confirmation missing (only qualified signature available)

In case of validation of document with mandant *upreg-fn,* overall result cannot be valid. E.g. qualified signature is valid, but additional validation rules apply to have a valid result.

Request:

```
validate.bat -c -u https://localhost:8080/validator/rs -m upreg-fn -f
examples\qualified.pdf -d -e
```

Result:

Only signature is valid:

```
results for signature with name: Signature1
Name of check: Signature status: VALID
Name of check: Certificate status: VALID
Name of check: Revocation status: VALID
Name of check: Timestamp status: VALID
Name of check: Mandant status: INVALID
```

Overall validation is invalid:

```
Validity of file report: INVALID
```

6.4 Document has been amended after applying function confirmation

In case a PDF document has been amended after the application of a function confirmation, the document content could have been changed.

From version 2.3.0, the library distinguishes between allowed and forbidden changes.

Requests:

```
validate.bat -c -u https://localhost:8080/validator/rs -m upreg-fn -f
examples\qualified-with-eobv-modified.pdf -d –e
```

Result:

Both signatures are valid, in terms of characteristics and number of signatures too. Nevertheless, status of file is negative because file has been amended after confirming the function.

```
Validity of file report: INVALID
was the document modified after last signature?: true
mandator requirements not met?: false
```

6.5 Last Signature is not confirmation from registry of deed

In case a PDF file is signed again after the confirmation of registry of deed, the validity of the file is not confirmed and should not be accepted.

Requests:

```
validate.bat -c -u https://localhost:8080/validator/rs -m upreg-fn -f
examples\qualified-with-eobv-with-qualified.pdf -d –e
```

Result:

All individual signatures are valid but still overall status of document is negative since the main requirement of EÖBV is not confirmed.

```
Signature Signature1 is VALID
Signature RegSig is VALID
Signature Signature2 is VALID
Validity of file report: INVALID
was the document modified after last signature?: false
mandator requirements not met?: true
```

6.6 Unsigned file is validated

An unsigned file can be validated without receiving an exception. Unsigned file results in negative validation result. In order to have a negative validation result and not an exception, Option *-e* must be indicated in CLI call.
New API has been extended with flag *processUnsignedFiles*.

Requests:

```
validate.bat -c -u https://localhost:8080/validator/rs -m upreg-fn -f
```

```
examples\unsigned.pdf -d -e
```

Result:

An unsigned file can now be validated. This always leads to a negative result, but no longer to an exception.
Can be switched on via the CLI using the option "-e"; without specifying the option, the old behavior is still executed.
The existing API has been extended by the optional flag "processUnsignedFiles".

Request:

```
validate.bat -c -u https://localhost:8080/validator/rs -m upreg-fn -f
examples\unsigned.pdf -d -e
```

Result:

```
Validity of file report: INVALID
```

7 Connection configuration

7.1 Test environment

The test environment of the discrete validator is available at the following URL:

Backend:

https://egovsigval-backend-d.bit.admin.ch

https://egovsigval-backend-d.bit.admin.ch/service/v3

7.2 Acceptance environment

The acceptance environment of the discrete validator is available at the following URL:

Backend:

https://egovsigval-backend-a.bit.admin.ch

https://egovsigval-backend-a.bit.admin.ch/service/v3

7.3 Production environment

The production environment of the discrete validator is available at the following URL:

Backend:

https://egovsigval-backend.bit.admin.ch

https://egovsigval-backend.bit.admin.ch/service/v3

8 Common Validator Mandants

Name of the mandants:	Purpose
FullQualified *(please use the mandant "Qualified")*	This validator checks whether a document is validly signed **with a qualified certificate and a qualified time stamp** of a recognized provider according to ZertES. All signatures contained in the document must meet these criteria. This client is used for documents that were signed after the revision of the ZertES, i.e. after 01 Jan. 2017.
Qualified	his validator checks whether a document is validly signed **with a qualified certificate** and a time stamp from a recognized provider in accordance with ZertES. The presence of a valid time stamp proving the exact time of signature is not necessary for a positive validation for documents signed before 1.1.2017 (entry into force of the revised Federal Act on Electronic Signature ZertES). All signatures contained in the document must comply with these criteria.
upreg-fn	This validator checks whether an electronic copy of a public deed or an electronic notarial certified copy in accordance with the Ordinance on Electronic Public Authentication (Verordnung über die elektronische öffentliche Beurkundung, EÖBV) is signed by a notary authorized in accordance with the Register of Notaries, provided with the confirmation of authorization of the Register of Notaries (register signature) and, if applicable, with the confirmation of authorization of a cantonal register (currently only the cantons of VD and GE). Both the signature of the certifying officer and the register signatures must be provided with a valid time stamp. Since the register signatures are linked to the signature of the notary, these signatures must always be validated together.
Siegel	This validator checks whether a document bears a regulated electronic seal according to ZertES. The presence of a qualified time stamp is necessary. All signatures contained in the document must meet these criteria.
Amtsblattportal	This validator whether the publications by a PDF/A-1a signed by SECO is. SECO has been operating the Official Gazette Portal (formerly SOGC) since autumn 2018. The official gazette portal (amtsblattportal.ch) can be used to record official publications that are published in the "Swiss Official Gazette of Commerce SOGC" (www.shab.ch) and/or in the cantonal official gazettes of Zurich (amtsblatt.zh.ch) and Basel-Stadt (kantonsblatt.ch).
edec	This validator checks whether the qualified signed document is a valid assessment ruling or a valid refund document from the Federal Customs Administration.
Strafregisterauszug	This validator checks whether a document is a valid Swiss criminal record extract.
eSchKG	This validator checks whether a document has been signed by a debt enforcement office. Such documents are sent by debt collection offices to participants with an eSchKG network.
FederalLaw	This validator checks whether there is a validly signed document on the federal publication platform (www.bundesrecht.admin.ch).

Name of the mandants:	Purpose
Indeterminate	This validator is a technical client, which is used if the validator cannot clearly assign the document type (the client) due to the context-based validation (or the validation rules stored for this). Specific validation rules and reports are defined for this technical client.
Mixed	This validator is a technical client which is used if the document to be validated has different types of electronic signatures, e.g. a qualified signature (QES) and one or more advanced signatures (FES), or a QES and a regulated seal, etc. The validator can also be used as a validation client. Specific validation rules and reports are defined for this technical client.
SwissGov-PKI	This validator checks whether a document has been signed with an advanced certificate on a Swiss Government PKI smartcard (FES) and provided with a time stamp from a recognized provider in accordance with ZertES. All signatures contained in the document must be valid and meet these criteria.
Kanton-Zug-Finanz-direktion	This validator checks whether a document has been validly signed by an administrative authority of the canton, the communes or the administrative court of the canton of Zug.
RegulatedSignature	This validator checks whether a document has been signed with a regulated certificate with the OID-policy 1.3.6.1.4.1.8024.1.300 issued by ‘QuoVadis Swiss Regulated CA G3’ when the remark is set to ‘regulated certificate’ as mentioned in the TAV.

Further clients are customer-specific and can be requested according to the procedure in Chapter 5.1 Support.

9 Sources

9.1 Discrete Validator – SDK & CLI

DE	→	https://www.bit.admin.ch/de/diskreter-validator-web-service
FR	→	https://www.bit.admin.ch/fr/le-service-en-ligne-du-validateur-discret
IT	→	https://www.bit.admin.ch/it/servizio-online-del-validatore-discreto
EN	→	https://www.bit.admin.ch/en/discreet-validator-web-service

9.2 Open eGov Tech Wiki

https://www.e-service.admin.ch/wiki/display/openegovdoc/Diskreter+Validator