# X.509 verifier, RESTful Web Services

## *[EN] User Manual*

"X.509 verifier" is a RESTful Web service that can serve to validate the X.509 certificates and signed PDF files as well as to check compliance of the contents of certificates with the Republic of Serbia legislative.

"X.509 verifier" RESTful Web service consists of 2 REST APIs located on the host:
http://signatureverifier.d-logic.com
and the paths of the scripts are:

1. /x509-verifier.php
2. /pdf-sgn-verifier.php

## REST API: x509-verifier

### API version: 1.0

PEM file, whose contents must be an X.509 certificate of version 3, is sent to this API. After checking certificate, the API returns result of the verification in the JSON encoded string.

### *HTTP server request*

**Host + Path:** http://signatureverifier.d-logic.com/x509-verifier.php
**Method:** POST
**Headers (mandatory):**
Content-Type: multipart/form-data; boundary=RANDOM_STRING_BOUNDARY
**Body:**

--RANDOM_STRING_BOUNDARY
Content-Disposition: form-data; name="file"; filename="file_name.pem"
Content-Type: application/octet-stream

[FILE_BINARY_DATA]
--RANDOM_STRING_BOUNDARY
Content-Disposition: form-data; name="query"

[JSON_ENCODED_PARAMETERS]
--RANDOM_STRING_BOUNDARY--
{END}

## *Description of the HTTP server request*

**RANDOM_STRING_BOUNDARY** is a string which needs to have a different and, if possible, a unique value on every new request. For example, in JavaScript, good practice for acquiring RANDOM_STRING_BOUNDARY would be:

```
var boundary = Math.random().toString().substr(2);
```

**[FILE_BINARY_DATA]** is a binary content of the chosen 'file_name.pem' file.

**[JSON_ENCODED_PARAMETERS]** are JSON encoded parameters which have to fulfil following format:

```
{
   "operation": "verify",
   "user_id": 123,
   "security_token": ""
}
```

and the good practice is to this JSON encoded string don't contains whitespace characters i.e. to be formed, in JavaScript, for example, using the following code:

```
var params, json;
params= { operation: "verify", user_id: 123, security_token: "" };
json = JSON.stringify(params);
```

Parameters are:

**"operation": "verify"** - Operation "verify" is the only operation currently supported.

**"user_id": 123** - numeric parameter, integer type, represent the user identification number (it is not utilised in API version 1.0 but it is mandatory and reserved for the future use). In API version 1.0 can be 0.

**"security_token": ""** - string which should contain pairs of the hexadecimal digits without a so called delimiters (it is not utilised in API version 1.0 but it is mandatory and reserved for the future use). In API version 1.0 can be an empty string.

Anyway, in JavaScript it is not necessary to directly manage the Content, i.e. HTTP body. We recommend using of the FormData class as in example which you can download from the git repository on the following URL:
https://www.d-logic.net/code/nfc-rfid-reader-sdk/signature_verifier_jc_example.git

There are also examples how to use cURL support from PHP to send requests to these REST APIs:
https://www.d-logic.net/code/digital_signature_sdk/php_example.git

## *HTTP Server Response*

After X.509 certificate verification, server will return JSON encoded string which (in API version 1.0) contains 2 arguments:

```
{"status":"STATUS_STRING","msg":"MESSAGE_STRING"}
```

On invalid request, server response will be:

```
HTTP/1.1 200 OK
…
Content-Type: application/json

{"status":"Error: wrong POST parameters.","msg":""}
```

If the verification is **successful**, STATUS_STRING must be:

```
"OK"
```

while the MESSAGE_STRING will contain a validly formatted record, which contains html formatting tags, as well as html tags for a new line, so this message can be directly placed in any html container (eg ).

Any response whose STATUS_STRING is different from "OK" is counted as a check whose result is **unsuccessful** and, in this case, if STATUS_STRING is different from the "Error: wrong POST parameters.", MESSAGE_STRING will contain details of the certificate scrutiny that should be displayed.

# REST API: pdf-sgn-verifier

### API version: 1.0

PDF file, whose contents must be signed PDF document, is sent to this API. Signature formats can be "PKCS#7 - Detached" or "CAdES Equivalent". After checking signed file, the API returns result of the verification in the JSON encoded string.

## *HTTP server request*

**Host + Path:** http://signatureverifier.d-logic.com/pdf-sgn-verifier.php
**Method:** POST
**Headers (mandatory):**
Content-Type: multipart/form-data; boundary=RANDOM_STRING_BOUNDARY
**Body:**

--RANDOM_STRING_BOUNDARY
Content-Disposition: form-data; name="file"; filename="file_name.pdf"
Content-Type: application/pdf

[FILE_BINARY_DATA]
--RANDOM_STRING_BOUNDARY
Content-Disposition: form-data; name="query"

[JSON_ENCODED_PARAMETERS]
--RANDOM_STRING_BOUNDARY--
{END}

## *Description of the HTTP server request*

**RANDOM_STRING_BOUNDARY** is a string which needs to have a different and, if possible, a unique value on every new request. For example, in JavaScript, good practice for acquiring RANDOM_STRING_BOUNDARY would be:

```
var boundary = Math.random().toString().substr(2);
```

**[FILE_BINARY_DATA]** is a binary content of the chosen 'file_name.pdf' file.

**[JSON_ENCODED_PARAMETERS]** are JSON encoded parameters which have to fulfil following format:

```
{
   "operation": "verify",
   "user_id": 123,
   "security_token": ""
}
```

and the good practice is to this JSON encoded string don't contains whitespace characters i.e. to be formed, in JavaScript, for example, using the following code:

```
var params, json;
params= { operation: "verify", user_id: 123, security_token: "" };
json = JSON.stringify(params);
```

Parameters are:

**"operation": "verify"** - Operation "verify" is the only operation currently supported.

**"user_id": 123** - numeric parameter, integer type, represent the user identification number (it is not utilised in API version 1.0 but it is mandatory and reserved for the future use). In API version 1.0 can be 0.

**"security_token": ""** - string which should contain pairs of the hexadecimal digits without so called delimiters (it is not utilised in API version 1.0 but it is mandatory and reserved for the future use). In API version 1.0 can be an empty string.

Anyway, in JavaScript it is not necessary to directly manage the Content, i.e. HTTP body. We recommend using of the FormData class as in example which you can download from the git repository on the following URL:
https://www.d-logic.net/code/nfc-rfid-reader-sdk/signature_verifier_jc_example.git

There are also examples how to use cURL support from PHP to send requests to these REST APIs:
https://www.d-logic.net/code/digital_signature_sdk/php_example.git

## *HTTP Server Response*

After verification of the PDF file and contains signature, server will return JSON encoded string which (in API version 1.0) contains 2 arguments:

```
{"status":"STATUS_STRING","msg":"MESSAGE_STRING"}
```

On invalid request, server response will be:

```
HTTP/1.1 200 OK
...
Content-Type: application/json

{"status":"Error: wrong POST parameters.","msg":""}
```

If the verification is **successful**, STATUS_STRING must be:

```
"PDF Signature is VALID"
```

while the MESSAGE_STRING will contain a validly formatted record, which contains html formatting tags, as well as html tags for a new line, so this message can be directly placed in any html container (eg ).

In contrary to the x509-verifier API, here we have responses with STATUS_STRING different from the "PDF Signature is VALID", which counted as a **unsuccessful** verification result but in these cases **MESSAGE_STRING doesn't contain details**. These are the cases when:

```
STATUS_STRING = "Error: PDF was changed after signing!"
STATUS_STRING = "Error: Wrong PDF format (while searching signature data)"
STATUS_STRING = "Info: PDF file doesn't contain digital signature"
STATUS_STRING = "Error: Wrong PKCS#7 format (missing \"to be signed\" data)"
```

The case when it is

```
STATUS_STRING = "Digital signature validation FAILED"
```

meaning that is verification of the PDF file and contains signature **entirely completed** but the **result is unsuccessful.** In this case the MESSAGE_STRING **always contain details of the scrutiny that should be displayed.** In this case MESSAGE_STRING will contain a validly formatted record, which contains html formatting tags, as well as html tags for a new line, so this message can be directly placed in any html container (eg ).

# Appendix: “Restlet Client” - Google Chrome browser extension exported files:

The accompanying part of this manual also includes JSON files that contain profiles for the "Google Chrome" extension "Restlet Client". These files are x509-verifier.json and pdf-sgn-verifier.json and contain profiles for REST APIs x509-verifier and pdf-sgn-verifier respectively. They can be used after installing "Restlet client" "Google Chrome" extensions.

“Restlet Client” have known bug which manifests itself by always, on profile loading, change the body form parameter type from “File” to “Text”. The workaround is to switch changed type back to “File” and then to chose the wanted file.

# Appendix: Change log

| **Date** | **Description** | **revision** | **refers to the API versions** x509-verifier / pdf-sgn-verifier |
|---|---|---|---|
| 2019-03-12 | • Translated two more sentences which were still in Serbian.<br>• Changed URLs from internal Gitlab to public Gitlab repository | 1.3 | 1.0 / 1.0 |
| 2019-01-25 | Translation to English. | 1.2 | 1.0 / 1.0 |
| 2019-01-19 | • Dodate su reference na PHP cURL primere u okviru odeljaka “Opis zahteva koji se šalje HTTP serveru”.<br>• Promenjen naslov ‘Appendix: “Restlet Client” - Google Chrome browser extension exported files’ zbog nejasnoća u vezi kopiranja JSON listinga iz dokumenta i “pejstovanja” u JSON fajlove. JSON | 1.1 | 1.0 / 1.0 |

| | | | |
|---|---|---|---|
| | fajlovi su prateći deo ovog dokumenta i moraju se nalaziti u istom folderu. | | |
| 2019-01-10 | First edition | 1.0 | 1.0 / 1.0 |