European Union
Regional Development Fund

Investing in your future

X-Road: Signed Document Download and Verification Manual

User guide

Version: 1.4

02.05.2016

10 pages

Doc. ID: UG-SIGDOC

Date	Version	Description	Author
18.06.2015	0.1	Initial version	Ilja Kromonov
03.07.2015	0.2	Corrections	Kristo Heero
19.08.2015	0.3	References updated	Kristo Heero
09.09.2015	1.0	Editorial changes made	Imbi Nõgisto
17.09.2015	1.1	Download URL of the asicverifier added	Kristo Heero
25.09.2015	1.2	„Forcing missing timestamps” added	Ilja Kromonov
15.02.2016	1.3	Parameter `subsystemCode` is mandatory also	Kristo Heero
02.05.2016	1.4	Signed documents are available by default 30 days.	Kristo Heero

Table of Contents

License

1 Introduction

Messages exchanged between the X-Road security servers are signed and encrypted. For every regular request and response, the security server produces a signed document. The signatures are time-stamped either synchronously, immediately after the message has been processed by the security server, or asynchronously in periodic batches for a better performance.

This document describes the retrieving and verification process of the signed and time-stamped document [PR-SIGDOC].

1.1 References

1. [PR-SIGDOC] Freudenthal, M. Profile for High-Perfomance Digital Signature. T-4-23, 2015. (http://cyber.ee/en/research/publications/research-reports/)

2 Signed Document Download Service

The security server offers the *asic* web service for downloading its signed documents. The service is used via HTTP GET requests to the service URL

```
http://SECURITYSERVER/asic
```

where `SECURITYSERVER` is the actual address of the security server.

Signed documents are available via the service until they are archived and removed from the message log database (by default 30 days). This time period is configurable in the security server (parameter *keep-records-for* in the configuration file */etc/xroad/conf.d/addons/message-log.ini*).

2.1 Retrieving Signed Documents of the Entire Transaction

The service requires the identifier of the corresponding message and the X-Road client identifier to determine which signed documents to return. These can be provided with the following mandatory parameters:

- *queryId* – the X-Road message transaction identifier, which is a part of the SOAP message header;
- *xRoadInstance* – the X-Road instance of the client identifier;
- *memberClass* – the member class of the client identifier;
- *memberCode* – the member code of the client identifier;
- *subsystemCode* – the subsystem code of the client identifier.

Thus, in order to retrieve the signed document for a message with transaction identifier *abc12345* exchanged by security server *sec1.gov* by client *EE/ENT/CLIENT1/SUB*, the request URL is

```
http://sec1.gov/asic?&queryId=abc12345
&xRoadInstance=EE&memberClass=ENT&memberCode=CLIENT1&subsystemCode=SUB
```

If a message with the given identifier was indeed exchanged by the security server and by the specified client, the server would respond with a ZIP archive (content-type `application/zip`, filename `queryId.zip`), which contains signed documents for all requests and responses that match the specified parameters.

The signed documents provided by the *asic* service are named `queryId-request-Z.asice` and `queryId-response-Z.asice` for requests and responses, respectively, where `queryId` is the identifier (URL encoded) of the message and `Z` is a 10-character random alphanumeric string.

2.2 Retrieving a Single Signed Document

Should the user only desire the request or response then additional mutually exclusive parameters are available:

- *requestOnly* – only include signed documents for request messages (response filename is `queryId-request.zip`);

- *responseOnly* – only include signed documents for response messages (response filename is `queryId-response.zip`).

The aforementioned parameters make the service return a ZIP archive, which may contain either one or more signed documents (depending if the provided `queryId` is unique). If only a single signed document is expected then the request can be further be constrained by adding the following parameter:

- *unique* – specifies that the only a single signed document is expected in the response, must be used in combination with either *requestOnly* or *responseOnly* parameter.

If this parameter is used and, indeed, the query identifier is unique, then the security server responds with a single signed document (content-type `application/vnd.etsi.asic-e+zip`) which represents the corresponding message.

2.3 Forcing Missing Timestamps To Be Created

If the security server was configured to timestamp messages asynchronously it may be possible that there is no timestamp associated with one or more signatures of the requested documents. In this case the request will fail with an internal error and return an appropriate error message string - "Some message signatures have not been timestamped yet!" (or "Message signature has not been timestamped yet!" if requesting a single document only).

If this behavior is not desired the following parameter can be used:

- *force* – specifies that the timestamping procedure should be executed explicitly in case any of the requested document signatures have no associated timestamps.

Thus, in order to retrieve the signed document for a message with transaction identifier *abc12345* exchanged by the security server *sec1.gov* by the client *EE/ENT/CLIENT1/SUB* and force any missing timestamps to be created, the request URL is

```
http://sec1.gov/asic?&queryId=abc12345
&xRoadInstance=EE&memberClass=ENT&memberCode=CLIENT1&subsystemCode=SUB&force
```

Should there be no working time-stamping provider available to the security server, the signed document retrieval service will respond with the error message "Failed to get timestamp from any time-stamping providers".

2.4 Authentication

In case the security server administrator has configured the connection between the service client and the security server to require authentication, requests to the *asic* service would need to be made via HTTPS.

The security server would need the certificate of the service client to be provided as part of the session, when the user makes the request to download a signed document for a message associated with this service client.

2.5 Error Conditions

The *asic* service responds with the HTTP error code and plain text error message if error occurs. The possible error codes are:

1. *400 Bad Request* – the combination of received parameters is invalid or some required

parameter is missing.

2. *401 Unauthorized* – the connection was made via HTTPS and the user failed to provide the correct certificate.
3. *404 Not Found* – no signed documents matching the provided parameters were found. Also, the signed document may be not time-stamped yet.
4. *500 Internal Server Error* – an unexpected internal error occurred (e.g., the provided service client identifier does not match any registered client on the security server).

3 Signed Document Verification Tool

Verification of signed documents is done by the *asicverifier* utility tool. The tool is written in the Java programming language and therefore requires Java 8 Runtime Environment (JRE) to be installed on the user's workstation. On Unix-like operating systems the JRE can be installed using package management software or downloaded from the Oracle website[1].

The *asicverifier* utility can be download from URL http://x-road.eu/packages/asicverifier-1.0.jar

3.1 Usage

The *asicverifier* utility is run as follows:

```
java -jar asicverifier-1.0.jar <configuration path> <signed document>
```

where `<signed document>` is the path to the signed document being verified and `<configuration path>` is the path to the verification configuration for this container (see Section 3.2).

If verification is successful the output will be similar to:

```
Loading configuration from18 verificationconf/...
Verifying ASiC container "abc12345-request-1ab2c3d4f5.asice" ...
Verification successful.
Signer
    Certificate:
        Subject: CN=CLIENT1, O=COM, C=EE
        Issuer: C=EE, O=EJBCA Sample, CN=AdminCA1
        Serial number: 897779140320284054
        Valid from: Mon May 04 14:30:38 EEST 2015
        Valid until: Wed May 03 14:30:38 EEST 2017
    ID: MEMBER:EE/COM/CLIENT1
OCSP response
    Signed by:
        Subject: C=EE, O=EJBCA Sample, CN=AdminCA1
        Issuer: C=EE, O=EJBCA Sample, CN=AdminCA1
        Serial number: 6005434255669835317
        Valid from: Thu Jun 14 13:04:29 EEST 2012
        Valid until: Sun Jun 12 13:04:29 EEST 2022
    Produced at: Fri Jun 05 08:47:11 EEST 2015
Timestamp
    Signed by:
        Subject: CN=timestamp1, O=Internet Widgits Pty Ltd, C=EE
        Issuer: CN=timestamp1, O=Internet Widgits Pty Ltd, C=EE
        Serial number: 12319570547049363959
        Valid from: Sun Nov 30 22:13:44 EET 2014
        Valid until: Wed Nov 27 22:13:44 EET 2024
    Date: Fri Jun 05 09:31:37 EEST 2015


Would you like to extract the signed files? (y/n) y
Created file message.xml
Files successfully extracted.
```

As can be seen from the example above, the *asicverifier* tool will optionally extract the

1 See http://www.oracle.com/technetwork/java/javase/downloads/index.html.

signed files to the working directory.

Should the verification fail, the reason for failure will be presented to the user in an error message. For example:

```
Loading configuration from verificationconf/...
Verifying ASiC container "abc12345-request-1ab2c3d4f5.asice" ...
Verification failed: Certificate is not issued by approved certification
service provider.
```

3.2 Verification Configuration

The *asicverifier* tool requires the proper verification configuration containing certificates needed by the verification process. The verification configuration can be downloaded from the same security server by making a HTTP GET request to the URL

```
http://SECURITYSERVER/verificationconf
```

where `SECURITYSERVER` is the actual address of the security server.

The *verificationconf* service has no parameters and responds with a ZIP archive (content-type *application/zip*, filename *verificationconf.zip*). This archive needs to be extracted and the location of it's contents used as an argument to the *asicverifier* tool.

A convenient way to download the verification configuration is with the web browser or use of the *curl* tool:

```
curl -J -O http://sec1.gov/verificationconf
```