European Union
Regional Development Fund

Investing in your future

X-ROAD 6

SIGNER CONSOLE USER GUIDE

2.5

VERSION HISTORY

Date	**Version**	**Description**
20.11.2014	0.1	First draft
20.11.2014	0.2	Some improvements done
1.12.2014	1.0	Minor corrections done
19.01.2015	1.1	License information added
02.04.2015	1.2	“sdsb” changed to “xroad”
30.06.2015	1.3	Minor corrections done
09.09.2015	2.0	Editorial changes made
14.09.2015	2.1	Audit log added
20.09.2015	2.2	Editorial changes made
06.09.2015	2.3	Added certificate request format argument
03.11.2015	2.4	Added label parameter for key generation command
10.12.2015	2.5	Editorial changes made

TABLE OF CONTENTS

1 INTRODUCTION

The purpose of this document is to explain how to manage keys and certificates in signer from the command line using the signer console utility.

Signer is an X-Road component whose primary purpose is to process signing requests and produce signatures. Signer manages software and hardware (smart cards, HSMs) tokens and their keys and handles certificates associated with the keys.

Signer console is a command line utility for interacting with Signer. The utility provides commands for various operations and can execute a single command or be started as an interactive text-based shell.

1.1 REFERENCES

1. [SPEC-AL] X-Road: Audit log events. Cybernetica AS.
2. [JSON] Introducing JSON, http://json.org/

2 USING THE SIGNER CONSOLE

2.1 SIGNER CONSOLE OPTIONS

Signer console accepts the following options:

```
-h or -help           displays list of supported commands
-v or -verbose        displays more detailed execution output
```

2.2 STARTING AS INTERACTIVE SHELL

To start the signer console as an interactive shell, type

```
sudo -u xroad -i signer-console [<options>]
```

2.3 EXECUTING SINGLE COMMANDS

To execute a single command, type

```
sudo -u xroad -i signer-console [<options>] <command> [<command arguments>]
```

2.4 AVAILABLE COMMANDS

This section gives an overview of all available commands in signer console.

A command may have one or more arguments, and may or may not produce any output. If command has arguments, they are always mandatory.

2.4.1 list-tokens

Description: Lists all tokens.

Arguments: (none)

Output: List of all tokens, each line representing the following token information:

```
<id> (<status>, <read only|writable>, <available|unavailable>, <active|
inactive>)
```

2.4.2 list-keys

Description: Lists all keys on all tokens.

Arguments: (none)

Output: List of keys on all tokens, each line representing the following key information:

```
<id> (<usage>, <available|unavailable>)
```

2.4.3 list-certs

Description: Lists all certificates and certificate requests on all keys.

Arguments: (none)

Output: List of certificates and certificate requests on all keys, each line representing the following information:

```
<id> (<status>, <client id>)
```

2.4.4 set-token-friendly-name

Description: Sets friendly name to the specified token.

Arguments:

- **token id:** the identifier of the token. Use **list-tokens** to look up token id-s.
- **friendly name:** the name to set.

Output: (none)

2.4.5 set-key-friendly-name

Description: Sets friendly name to the specified key.

Arguments:

- **key id:** the identifier of the key. Use **list-keys** to look up key id-s.
- **friendly name:** the name to set.

Output: (none)

2.4.6 get-member-certs

Description: Returns certificates of a member.

Arguments:

- **member id:** the identifier of the member, entered as

  `\"<instance> <class> <code>\".`

Output: List of certificates of the specified member.

2.4.7 activate-certificate

Description: Activates the specified certificate.

Arguments:

- **certificate id:** the identifier of the certificate. Use **list-certs** to look up certificate identifiers.

Output: (none)

2.4.8 deactivate-certificate

Description: Deactivates the specified certificate.

Arguments:

- **certificate id:** the identifier of the certificate. Use **list-certs** to look up certificate identifiers.

Output: (none)

2.4.9 delete-key

Description: Deletes the specified key and all associated certificates and certificate requests.

Arguments:

- **key id:** the identifier of the key. Use **list-keys** to look up key id-s.

Output: (none)

2.4.10 delete-certificate

Description: Deletes the specified certificate from Signer.

Arguments:

- **certificate id:** the identifier of the certificate. Use **list-certs** to look up certificate identifiers.

Output: (none)

2.4.11 delete-certificate-request

Description: Deletes the specified certificate request from Signer.

Arguments:

- **certificate request id:** the identifier of the certificate. Use **list-certs** to look up certificate request identifiers.

Output: (none)

2.4.12 import-certificate

Description: Imports a certificate from the specified file to a key. The certificate is imported to the key pair whose public key matches that of the certificate.

Arguments:

- **file:** the relative or absolute file name of the certificate in PEM or DER format.
- **member id:** the identifier of the member constructed from the C, O, CN fields of the certificates DN, entered as:

```
\"<instance> <class> <code>\".
```

If the certificate is an authentication certificate, the member id is not used.

Output: Identifier of the key to which the certificate was imported.

2.4.13 login-token

Description: Log in to the specified token.

Arguments:

- **token id:** the identifier of the token. Use **list-tokens** to look up token identifiers.

Output: (none)

2.4.14 logout-token

Description: Log out of the specified token

Arguments:

- **token id:** the identifier of the token. Use **list-tokens** to look up token identifiers.

Output: (none)

2.4.15 init-software-token

Description: Initialize the software token. A PIN is prompted that is used to log in to this token.

Arguments: (none)

Output: (none)

2.4.16 sign

Description: Sign the specified character data using the specified key.

Arguments:

- **key id:** the identifier of the key. Use **list-keys** to look up key identifiers.
- **data:** character data to be signed as string

Output: signature byte array

2.4.17 generate-key

Description: Generates a key on the specified token.

Arguments:

- **token id:** the identifier of the token. Use **list-tokens** to look up token identifiers.
- **label:** the label of the key is set for SSCD devices.

Output: The id of the generated key.

2.4.18 generate-cert-request

Description: Generates a certificate request under the specified key in Signer and saves it to a CSR file in current directory.

Arguments:

- **key id:** the identifier of the key. Use **list-keys** to look up key identifiers.
- **member id:** the identifier of the member that matches the subject name, entered as:

  ```
  \"<instance> <class> <code>\".
  ```
- **usage:** key usage – either „`s`" (sign) or „`a`" (authentication)
- **subject name:** the subject distinguished name, entered as:

  ```
  „C=<instance>, O=<class>, CN=<code>"
  ```
- **format:** the format of the generated certificate request – either „`der`" or „`pem`"

Output: Name of the file where the certificate request was saved.

3 EXAMPLE: CERTIFICATE IMPORT

The following usage example shows how to initialize a software token and import a certificate to signer.

1. Initialize the software token:

   ```
   signer-console init-software-token
   ```

   A PIN is prompted, this will be used to log in to the software token afterwards.

2. Log in to the software token:

   ```
   signer-console login-token 0
   ```

   Note, that the identifier of software token is always „0”.

3. Generate a new key on the software token:

   ```
   signer-console generate-key 0
   ```

   Output is key id:

   ```
   F30D41B745FC072028956A3E9695416247248595
   ```

4. Create a certificate request:

   ```
   signer-console generate-cert-request
   F30D41B745FC072028956A3E9695416247248595 \”FOO BAR BAZ\” s
   ”C=FOO,O=BAR,CN=BAZ” pem
   ```

   Output:

   ```
   Saved to file F30D41B745FC072028956A3E9695416247248595.csr
   ```

5. Send the CSR to the Certificate Authority and get the certificate.

6. Import the new certificate to signer:

   ```
   signer-console import-certificate <PEM file> ”SAVED” \”FOO BAR BAZ\”
   ```

4 AUDIT LOG

User actions events that are made by the signer-console utility and that change the system state or configuration are logged to the audit log. The actions are logged regardless of whether the outcome was a success or a failure. The complete list of the audit log events is described in [SPEC-AL].

An audit log record contains

- the description of the user action,
- the date and time of the event,
- the user name of the user performing the action, and
- the data related to the event.

For example, logging in to the token produces the following log record:

```
2015-09-14T17:41:28+03:00 my-server-host INFO  [X-Road Signer Console] 2015-
09-14 17:41:28+0300 - {"event":"Log into the token","user":"xroad","data":
{"tokenId":"0"}}
```

The event is present in JSON [JSON] format, in order to ensure machine processability. The field `event` represents the description of the event, the field `user` represents the user name of the performer, and the field `data` represents data related with the event. The failed action event record contains an additional field `reason` for the error message. For example:

```
2015-09-14T17:43:07+03:00 my-server-host INFO  [X-Road Signer Console] 2015-
09-14 17:43:07+0300 - {"event":"Log into the token
failed","user":"xroad","reason":"Signer.PinIncorrect: PIN incorrect","data":
{"tokenId":"0"}}
```

By default, in the X-Road security server and central server, audit log is located in the file

```
/var/log/xroad/audit.log
```