

u.trust Anchor FIPS 140-3

csadm Manual



Imprint

Copyright 2024	Utimaco IS GmbH Germanusstr. 4 D-52080 Aachen Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226) EMEA +49 800-627-3081 APAC +81 800-919-1301
Internet e-mail	https://support.hsm.utimaco.com/ support@utimaco.com
Document Version	1.0.2
Product Version	6.0.0
Date	2024-10-25
Document No.	2023-0037
Status	PUBLISHED

All rights reserved	<p>No part of this documentation may be reproduced in any form (printing, photocopy or according to any other process) without the written approval of Utimaco IS GmbH or be processed, reproduced or distributed using electronic systems.</p> <p>Utimaco IS GmbH reserves the right to modify or amend the documentation at any time without prior notice. Utimaco IS GmbH assumes no liability for typographical errors and damages incurred due to them. Any mention of the company name Utimaco in this documents refers to the Utimaco IS GmbH.</p> <p>All trademarks and registered trademarks are the property of their respective owners.</p>
---------------------	--

Table of Contents

1	Introduction	7
1.1	Document Conventions	7
2	Installation	9
2.1	Optional Configuration	11
2.1.1	Setting, Verifying and Deleting CRYPTOSERVER on Windows	11
2.1.2	Setting, Verifying and Deleting CRYPTOSERVER on Linux	13
3	Security Management	15
3.1	Maximum for Failed Authentication Attempts	15
3.2	Secure Messaging	15
3.3	TLS connection to u.trust Anchor LAN	18
3.3.1	Environment variables	18
3.4	Auditing	19
3.4.1	Signed Audit Logs	20
3.4.2	Signed Configuration Files	20
3.4.2.1	Sample Configuration File cmds_sample.cfg	22
3.4.2.2	Interface Hardening by Disabling Selected Functions	22
3.4.2.3	Configurable Role-Based Access	23
3.4.2.4	List of External Firmware Functions	24
3.5	Security Guidelines	25
3.5.1	General Advice	25
3.5.2	Protected Operational Environment	25
3.5.3	Protection of Data Outside the u.trust Anchor	27
4	Administration of u.trust Anchor with csadm	28
4.1	Overview of csadm	28
4.1.1	Syntax of csadm	28
4.1.2	Command Execution	31
4.1.3	Password Authentication	33
4.1.4	Storage and Specification of RSA and ECDSA Keys for Authentication	34
4.1.4.1	Key Specifiers	35
4.1.4.2	Smartcard Specifiers	35
4.1.5	Availability of commands on FIPS cHSMs	37
4.2	Basic Commands	39
4.2.1	Cmd	39

4.2.2	2020-0037 CmdFile	40
4.2.3	ConnTimeout	42
4.2.4	Help	42
4.2.5	PrintError	43
4.2.6	SetTimeout	43
4.2.7	Sleep	44
4.2.8	Version	44
4.2.9	StrError	45
4.2.10	RamInfo	45
4.2.11	RamInfoCSV	46
4.2.12	GenRandom	47
4.3	Authentication Commands	47
4.3.1	LogonPass	49
4.3.2	LogonSign	51
4.3.3	ShowAuthState	53
4.4	Commands for Administration	54
4.4.1	GetState	54
4.4.2	SetAdminMode	56
4.4.3	SetStartupMode	57
4.4.4	GetStartupMode	58
4.4.5	GetTime	58
4.4.6	SetTime	59
4.4.7	ListFiles	60
4.4.8	LoadFile	61
4.4.9	DeleteFile	62
4.4.10	ListFirmware	63
4.4.11	GetBootLog	65
4.4.12	GetAuditLog	67
4.4.13	ClearAuditLog	67
4.4.14	GetAuditConfig	68
4.4.15	Test	68
4.4.16	SetAuditConfig	70
4.4.17	GetCertificates	71
4.4.18	LoadCertificate	72

4.4.19	GenerateAuditLogKey.....	72
4.4.20	GetAuditLogKey.....	73
4.4.21	GetInfo.....	74
4.4.22	GetBattState.....	75
4.4.23	GetSignedAuditLog.....	76
4.4.24	VerifySignedAuditLog.....	77
4.4.25	ClearAuditLogFiles	77
4.4.26	LoadAltMdlSigKey	78
4.4.27	BackupDatabase.....	80
4.4.28	RestoreDatabase	82
4.4.29	Restart.....	84
4.4.30	SignConfig	84
4.5	User Management Commands	86
4.5.1	AddUser	89
4.5.1.1	Adding PKCS#11 Security Officers with RSA Signature Authentication with a Smartcard.....	91
4.5.2	BackupUser	93
4.5.3	ChangeUser	95
4.5.4	DeleteUser	98
4.5.5	GetMaxAuthFails	99
4.5.6	ListUser	99
4.5.7	RestoreUser	101
4.5.8	SetMaxAuthFails	107
4.6	Commands for Managing the User Authentication Keys.....	108
4.6.1	BackupKey.....	109
4.6.2	CopyBackupCard	109
4.6.3	ChangePassword.....	110
4.6.4	ChangePIN.....	112
4.6.5	GetCardInfo	112
4.6.6	GenKey	113
4.6.7	SaveKey (Standard).....	117
4.7	Master Backup Key Commands.....	118
4.7.1	MBKCardCopy.....	120
4.7.2	MBKCardInfo	120
4.7.3	MBKPINChange	121
4.7.4	MBKImportKey	122

4.7.5	MBKGenerateKey	124
4.7.6	2020-0037 MBKListKeys	125
4.7.7	2020-0037 MBKCopyKey	126
4.8	Commands for Administration of u.trust Anchor LAN	127
4.8.1	CSLGetConnections	128
4.8.2	CSLGetVersion	129
4.8.3	CSLGetStatus	130
4.8.4	CSLGetLogFile	131
4.8.5	CSLGetConfigFile	132
4.8.6	CSLSetTracelevel	133
4.8.7	CSLShutdown	134
4.8.8	CSLReboot	135
4.8.9	CSLGetTime	136
4.8.10	CSLSetTime	137
4.8.11	CSLGetSerial	138
4.8.12	CSLGetLoad	139
5	Basic Administration Tasks	140
5.1	Entering and Leaving the 'Administration only' Mode	140
5.2	Generating a User Authentication Key	141
6	Advanced Administration Tasks	144
6.1	Generating and Verifying Signed Audit Log Files	144
6.2	Using the Signed Configuration File cmds.scf	146
7	Contact Address for Support Queries	149

1 Introduction

This document provides information about the command-line tool *csadm*. It is used to administer containerized Hardware Security Modules (cHSMs) of the u.trust Anchor product line. The tool handles all typical administration tasks, such as setup, status monitoring, managing users, firmware, and keys.

Additionally, it can perform advanced administration functions that are exclusively available for customers working with SDK devices that want to extend the standard functionality of the device with self-developed firmware modules providing specific cryptographic functions and commands.

All operating systems that are currently supported for the *csadm* host computer are listed in the release notes.

We hope you are satisfied with our product. Please do not hesitate to contact us if you have any questions or comments.

1.1 Document Conventions

We use the following document conventions:

Convention	Use	Example
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press OK
<code>Monospaced</code>	Code that is given for explanation or as an example, file paths	<code>chsm-create</code>
<i>Italic</i>	References and important terms	See <i>Sample Chapter</i> in the <i>CryptoServer - Sample Manual</i>

Table 1: Document conventions

We use special icons to highlight the most important notes and information.



Here, you find important safety information that should be followed.



Here, you find additional notes or supplementary information.



This message marks the result expected after the successful execution of an instruction.

2 Installation

Installing csadm on a computer with a Windows operating system

This section describes how to install csadm on a Windows administration computer. csadm is used to manage cHSMs.

On an administration computer running a Windows operating system, csadm is installed by default during the installation of the u.trust Anchor software provided in the product bundle.

The following steps describe how to install csadm on a Windows host computer.

The `csadm.exe` file for Windows can be found in the product bundle here:

- For Windows 32-bit operating systems
Not supported
- For Windows 64-bit operating systems
`Software\Windows\Administration\`

1. Copy the `csadm.exe` file to a well-chosen directory.
2. Add this directory to the `PATH` environment variable to be able to call the administration tool from any other directory.
3. If you do not want to set the `Dev=` parameter with each execution of a csadm command:
It is possible to set an environment variable `CRYPTOSERVER`, which sets the CryptoServer address permanently (unless a `Dev=` parameter is explicitly set for a specific command).



csadm has been successfully installed on the administration computer.

Installing csadm on a computer with a UNIX-like operating system

This section describes how to install csadm on an administration computer. csadm is used to manage cHSMs.

You find the `csadm` file in the product bundle here:

- For 32-bit operating systems
Not supported
- For 64-bit operating systems
`/Software/Linux/Administration/`

1. Create a `~/bin` directory in your user directory, if there is not one yet:

```
mkdir ~/bin
```

2. Copy the csadm relevant for your operating system into the `~/bin` directory. An example for Linux 64-bit:

```
cp <mount point of the product CD>/Software/Linux/Administration/csadm ~/bin
```

3. Ensure that you have write and execute permissions for csadm.

```
chmod -R u+w+x ~/bin
```

4. Add the `~/bin` directory to the path in the user configuration file in the shell that is being used. In this example, a bash is used as the shell, i.e., open the `~/.bashrc` file and add the following line to it:

```
export PATH=$PATH:~/bin
```

5. Save the changes and close the `~/.bashrc` file.



csadm has been successfully installed on the administration computer.

An environment variable CRYPTOSERVER can be set to avoid setting the `Dev=` parameter for every command execution (e.g., with the value `/dev/cs2.n.m` where `n = 0` and `m = {1, 2, 3, 4}` or `m = {1, 2 ..., 12}`). The CRYPTOSERVER variable defines the cHSM address permanently, but is ignored when a `Dev=` parameter is explicitly set for a specific command.

2.1 Optional Configuration

For setting up and administering the device, the command-line tool csadm requires the address of the device to which it will connect. You can set this address by specifying it for every command in the `Dev` command parameter.

Example:

```
csadm Dev=4001@192.168.1.1 GetState
```

Optionally and for more convenience, you can set the address of your device in the `CRYPTOSERVER` environment variable. In this case, the `Dev` parameter in the csadm command can be omitted.

The `CRYPTOSERVER` environment variable is not a default for the following parameters:

- `Device` parameter in the `cs_pkcs11_R3.cfg` file
This parameter is used for actions initiated by p11tool2. For details, see *CryptoServer – PKCS#11 p11tool2 Reference Manual*.
- `Device` parameter in the `csxlan.conf` file
This parameter is used for some communication of the u.trust Anchor LAN.

2.1.1 Setting, Verifying and Deleting CRYPTOSERVER on Windows

Setting CRYPTOSERVER

- For the current command-line
Example:

```
set CRYPTOSERVER=192.168.1.1
```
- For all future command-lines
 - Using an administrator command-line
 - **Start > All Programs > Accessories > Right mouse click on Command Prompt > Run as administrator**
 - Perform a setx command
Example:

```
setx CRYPTOSERVER 192.168.1.1 /m
```


The /m parameter indicates that the environment variable is a system environment

variable and no user environment variable. The value is only set for future command-lines but not for the current one.

- Using the GUI
 - Ensure that you are logged in as an administrator.
 - **Start > Control Panel > System and Security > System > Advanced system settings > Advanced > Environment variables > System variables > New** or select **CRYPTOSERVER > Edit** plus OK

Verifying CRYPTOSERVER

- For the current command-line
Example: `set CRYPTOSERVER`
- For all future command-lines
 - Ensure that you are logged in as an administrator.
 - **Start > Control Panel > System and Security > System > Advanced system settings > Advanced > Environment variables > System variables > CRYPTOSERVER**

Deleting CRYPTOSERVER

- For the current command-line
Example:
`set CRYPTOSERVER=`
- For all future command-lines
- Using an administrator command-line
 - **Start > All Programs > Accessories > Right mouse click on Command Prompt > Run as administrator**
 - Perform a setx
Example:
`setx CRYPTOSERVER "" /m`
The /m parameter indicates that the environment variable is a system

environment variable and no user environment variable. The value is only set for future command-lines but not for the current one.

- Using the GUI
 - Ensure that you are logged in as an administrator.
 - Start > Control Panel > System and Security > System > Advanced system settings > Advanced > Environment variables > System variables > CRYPTOSERVER > Delete

2.1.2 Setting, Verifying and Deleting CRYPTOSERVER on Linux



The commands in this section are only examples. Depending on your shell, other commands might be suitable.

Setting CRYPTOSERVER

- For the current command line

Example:

```
export CRYPTOSERVER=192.168.1.1
```

- For all future command-lines

- a. Append an export command to the `.bashrc`



It is very important that you use `>>` in the following command. It appends the preceding text to the specified file. Do not use `>`, because it overwrites the entire file.

Example:

```
echo export CRYPTOSERVER=192.168.1.1 >> ~/.bashrc
```

- b. Apply the changes to the current command-line as well by performing the following command.

Example: `. ~/.bashrc`

Verifying CRYPTOSERVER

Perform one of the following commands to verify the value of the CRYPTOSERVER environment variable.

- For the current command line

Example 1: `env | grep CRYPTOSERVER`

Example 2: `printenv | grep CRYPTOSERVER`

Example 3: `echo $CRYPTOSERVER`

Deleting CRYPTOSERVER

- For the current command-line

Example:

```
export CRYPTOSERVER=
```

- For all future command-lines

Remove the following line from the `~/.bashrc` file.

Example:

```
export CRYPTOSERVER=192.168.1.1
```

3 Security Management

3.1 Maximum for Failed Authentication Attempts

In the device, the number of consecutive failed authentication attempts for every user is automatically counted and stored as a user attribute (counter for failed authentication attempts, *AuthenticationFailureCounter*, denoted as $Z[n]$). If the authentication of a user fails, the *AuthenticationFailureCounter* for the user is incremented by one. On successful authentication the *AuthenticationFailureCounter* is reset to zero.

A maximum value for the number of failed authentication attempts (*MaxAuthFails*) can be set by using the csadm command `SetMaxAuthFails`, where $0 \leq \text{MaxAuthFails} \leq 255$. To retrieve the defined value for *MaxAuthFails* the csadm command `GetMaxAuthFails` should be used. By default, *MaxAuthFails* = 0 which means that the number of failed authentication attempts for all device users is not limited. Any other value for *MaxAuthFails* means that for each user there are only *MaxAuthFails*-1 consecutive failed authentication attempts allowed. The user is locked automatically after *MaxAuthFails* consecutive failed authentication attempts, i.e., if the *AuthenticationFailureCounter* of the user equals the value of *MaxAuthFails*.

If a user is locked no further authentication is possible for her/him, consequently this user cannot perform any command which has to be authenticated. Only a user with user management permission (permission 2 in user group 7, 20000000) can unlock a locked user by setting her/his *AuthenticationFailureCounter* back to zero by using the csadm command `ChangeUser`.

3.2 Secure Messaging

The device supports Secure Messaging for communications between the host and itself. Commands from the host to the device and the replies to the host can be AES encrypted and the integrity of the data can be protected by a AES MAC (Message Authentication Code). For this purpose, a secure messaging header data block is added to the command and the answer data block. The detailed structure of the header data blocks is described in *CryptoServer - Firmware Module CMDS - Interface Specification* provided within the product bundle.

In Secure Messaging between the device and the host, a new random session key is generated for each connection. This prevents recorded data packets belonging to an earlier connection from being imported into a new connection. No valid data packets can be present once the imported data packet has been decrypted with the current session key.

In addition, the device generates a start value for a sequence counter and a session ID for every new session key. The sequence counter increases every time a command is sent and is also included in the MAC calculation or integrity check. This ensures that no commands are recorded within the same connection and sent to the device again. The unique session ID

guarantees that every session key is uniquely identifiable. All commands that use the same session key and session ID are part of the same session (connection).

u.trust Anchor cHSMs support a maximum of 4096 session keys (connections) active at the same time. If another session key is requested for a connection, exceeding the maximum of 4096, the oldest connection is closed and the session key, associated with it, becomes invalid.



On FIPS devices, Secure Messaging must be used for every command which has to be authenticated.

To use the secure messaging functionality, the csadm commands `csadm LogonSign` and `csadm LogonPass`, are used together in one command line with the command(s) for which Secure Messaging should be used. csadm will automatically open an authenticated Secure Messaging session, perform the specific command(s) with Secure Messaging (i.e., encrypted and MAC-secured) and close the Secure Messaging session on the device again. The host and the device agree upon an AES session key. The external device function `GetSessionKey` is called to generate an AES session key.

1. The session key can be negotiated in one of the following ways:

- With the Ephemeral Unified Model Key Agreement Scheme according to SP800-56A r3 (FIPS-compliant; based on EC Diffie Hellman). This is automatically tried first. If the device does not support this, the Diffie-Hellman key establishment protocol is used.
- With the Diffie-Hellman key establishment protocol



On FIPS devices the Diffie-Hellman key establishment protocol is blocked.

2. The host and the device exchange encrypted commands in a Secure Messaging session. The host can now send commands to the device that are AES-encrypted and integrity-protected with an AES MAC. The respective answers to these commands that are sent back to the host by the device are always encrypted and protected with a MAC, too. The

sequence counter is used as initialization vector for the AES encryption and MAC calculation and is incremented after every command to prevent unauthorized replays of the commands.



The session key is identified by a session ID. All commands using the same session ID and the same session key belong to the same session. This way a secure channel can be established between the device and the host application using the Secure Messaging mechanism.

The device accepts commands that do not require authentication (i.e., in cleartext), sent to it over an unprotected connection, in parallel to commands sent to it within a protected Secure Messaging session. If the device receives an encrypted command (i.e., a command using the Secure Messaging layer of the protocol stack of the device), it checks the MAC. The command is rejected if the MAC is invalid.



A session key automatically becomes invalid if it is not used for 15 minutes. The device supports a maximum of 4096 session keys that can be active simultaneously and can be used by different host applications (each key identified by its session ID). If a host application requests a new session key while the maximum of 4096 sessions are already active, the oldest session is closed and the session key associated with it becomes invalid.

If the `GetSessionKey` function is authenticated by one or more users using single command authentication, the permissions of the corresponding user(s) will be granted to the established Secure Messaging session. All commands that are protected with this session key have the permissions of these users without any extra authentication being necessary. However, outside of this session, the former authentication status is preserved.



For the EC-Diffie-Hellman key establishment protocol, see *CryptoServer - PKCS#11 R3 - Developer Guide*.

3.3 TLS connection to u.trust Anchor LAN



TLS is only supported for Linux clients.

The TLS implementation for u.trust Anchor LAN allows the client tool `csadm` to connect to u.trust Anchor LAN over TLS and access the cHSM instances.

You can select TLS by extending commands with a `TLS` specifier.

Example

```
csadm dev=TLS:4001@192.168.1.1 GetState
```

The connection to the devices is established using TLS.

The prerequisite is that TLS is correctly configured on u.trust Anchor LAN and on the administration computer on which you run `csadm`:

- You have to set certain environment variables in the command shell in which you execute `csadm`, to use the desired certificates and keys to connect.
- For the configuration on u.trust Anchor LAN, see TLS for u.trust Anchor LAN in the *u.trust Anchor LAN V5 - Administration Manual*,


If you do not specify TLS, TCP is used by default.

For an example of configuring TLS, see *How to configure TLS* in the *u.trust Anchor LAN V5 - Administration Manual*.

3.3.1 Environment variables

For information on setting environment variables using the CRYPTOSERVER environment variable as an example, see Optional Configuration.

Variable	Description
TLS_CLIENT_CERT	Path to the file that holds the client certificate. Example: <code>export TLS_CLIENT_CERT=/UTI/SE5K/MTLS/HostCert.pem</code>

Variable	Description
TLS_CLIENT_PWD	Optional password for encrypted client certificate. Example: <code>export TLS_CLIENT_PWD=my_password</code>
TLS_CLIENT_KEY	Path to the file which holds the client private key. Example: <code>export TLS_CLIENT_KEY=/UTI/SE5K/MTLS/HostPrivkey.pem</code>
TLS_CA_BUNDLE	Path to the file which holds the CA certificate chain. Example: <code>export TLS_CA_BUNDLE=/UTI/SE5K/MTLS/CACert.pem</code>
TLS_SKIP_DN_VERIFY	If set it skips the domain name verification. The domain name of the server is compared to the DN or alternative subject name in the server certificate. Example: <code>export TLS_SKIP_DN_VERIFY=1</code>  Only use this functionality for a limited time for testing purposes or for troubleshooting.
TLS_VERBOSE	Show more output in case of errors. Example: <code>export TLS_VERBOSE=1</code>
TLS_DEBUG	Show maximum output for problem analysis. Example: <code>export TLS_DEBUG=1</code>



`TLS_VERBOSE` and `TLS_DEBUG` are only available on the client side. `TLS_VERBOSE` provides the ability to better see errors on the client side. With `TLS_DEBUG` you can analyze problems even more precisely. These options are not available on the u.trust Anchor LAN side. There errors are always written to the `csxlan.log` file.

3.4 Auditing

In the audit log all events and actions involving security issues that occur or are performed whilst the cHSM is running are recorded.



For a detailed overview of audit log entries, see *Audit Log Entries* in the *u.trust Anchor - Administration Manual*.

3.4.1 Signed Audit Logs

The CryptoServer can generate signed audit log files on the computer on which csadm is running.

The audit log signature key, which is described in *Audit Log Signature Key* in the *u.trust Anchor - cHSM - Administration Manual*, is used for this process. The generated files have extra long audit log file names, see *AuditLogFileNames* in the *u.trust Anchor - cHSM - Administration Manual*. For details about how to generate and use this key, see section [Generating and Verifying Signed Audit Log Files](#).

3.4.2 Signed Configuration Files

The device software supports the use of signed configuration files with the `.scf` file extension. In such a signed configuration file you can define specific firmware module configuration settings, for example, to disable selected device functions which enables the hardening of these functions, and to define elevated permission/authentication requirements for specific device functions.

The signed configuration files are protected with a signature using an Alternative Module Signature Key, and can only be loaded into the device by a user with system administrator permission (2 in the user group 6). Like this setting, changing or removing additional security rules defined in a signed configuration file requires the same permissions and protection as those for loading and changing the device firmware.

A signed configuration file is originally created as a configuration file with any file extension, for example, `*.cfg`, `*.txt`. This configuration file is then signed with the Module Signature Key or the Alternative Module Signature Key by using the csadm command `SignConfig`. The Alternative Module Signature Key is created by the customer outside the device and imported into it (by using the csadm commands `GenKey` and `LoadAltMdlSigKey`). The `.scf` file is loaded into the device with the csadm command `LoadFile`. While loading it into the device the signature of the `.scf` file is verified by the ADM firmware module. After successful signature verification and before the signed configuration file is stored on the device's file system, the signature is replaced by a SHA-512 hash value. On every startup this hash value is recalculated and compared with the previously stored one, in order to check the integrity of the signed configuration file.

Syntax

The syntax of a signed configuration file is as follows:

- Sections are defined as `[Section name]`.
- Global configuration items should be specified above the first section definition.
- Configuration items are defined as combination of a key and a value:
`<key> = <value_1>,<value_2>,...,<value_n>`

Rules

- Leading and trailing space characters are stripped on `<keys>` and `<values>`.
- A `<key>` should not contain any space characters.
- The character `#` should be used for commenting out a line.
- Lines with an invalid syntax are ignored.
- One `[Section]` should be used only once in the current configuration file.
- A `<key>` should be used only once in the current `[Section]`.

If there are several `<key>` entries, only the first one applies. The others are ignored.

- The `<value>` might be split over several lines which should end with a backslash `"\"`.

Deleting and Replacing Signed Configuration Files

Signed configuration files are only deleted when:

- Executing the csadm command `Clear`
- Loading a new firmware package into the device (e.g. by using the csadm command `LoadPkg` with parameter `ForceClear`)
- Executing a Clear-to-Factory-Defaults by performing an External Erase

Signed configuration files cannot be deleted by using the csadm command `DeleteFile`, and are not deleted when an alarm has occurred.

You can load a new signed configuration file into the device with the csadm command `LoadFile`. The existing one is then overwritten.

3.4.2.1 Sample Configuration File `cmds_sample.cfg`

Utimaco provides a sample configuration file `cmds_sample.cfg` in the product bundle in one of the following directories:

```
\Software\Windows\Administration
```

```
\Software\Linux\Administration
```

It contains a list of the device firmware modules, providing external interfaces, and their unique module IDs (function code (FC)), as well as complete list of their external functions/interfaces (subfunction codes (SFC)). You can use this sample configuration file as a basis for your own configuration/signed configuration file. After editing the sample configuration file to fit to your requirements, you should rename it to `cmds.cfg` so that it can be interpreted by the device.

3.4.2.2 Interface Hardening by Disabling Selected Functions

The device software offers the possibility to additionally secure the device interfaces by disabling selected device functions. These functions have to be defined in the signed configuration file `cmds.scf`.

For that purpose, the configuration file `cmds_sample.cfg` has to contain the dedicated section `[DisableSFC]` specifying which functions should be disabled. The syntax is as follows:

```
[DisableSFC]
<FC> = <SFC1>,<SFC2>,...,<SFCn>
or for better readability
<FC> = <SFC1>,\
    <SFC2>,\
    ..., \
    <SFCn>
```

FC is the function code (ID) of the firmware module, which is exactly three hex digits with leading 0x, e.g., "0x012".

SFC is the specific decimal sub-function code (ID), specifying a function within a firmware module, which is disabled. The SFCs do not have to be ordered numerically.

The disabled firmware module functions are listed during device startup in the Boot Log in specific entries with the syntax

```
CMD5/DSOM: <FC> - disabled <SFC1> <SFC2> ... <SFCn>.
```

For example, `CMD5/DSOM: 0x083 - disabled 0 1 17` means that in the firmware module CMD5 (with function code FC = 0x83) the functions Echo (with SFC = 0), Reverse Echo (with SFC = 1) and Set Maximum Authentication Failures (with SFC = 17) are disabled, and cannot be used by external applications.

If you try to access a disabled device function the following error message is returned by the device:

```
Error B0830061
```

This function is not available in this HSM configuration.

See [Using the Signed Configuration File `cmds.scf`](#) for step-by-step instructions on how to create/edit, sign, load and activate your signed configuration file `cmds.scf`.

3.4.2.3 Configurable Role-Based Access

Role-based access provides the possibility to increase the necessary authentication level for selected device functions, and thus, to even further restrict the access to the system to users with customized roles/permissions.

The CryptoServer maximal possible authentication status that can be reached for selected functions in the different user groups has been increased to 15 (0xF).

The increased permission level enabling specific users to execute specific functions can be individually configured by the customer for the device external functions with help of the signed configuration file `cmds.scf`.

In order to do so, the configuration file `cmds_sample.cfg` has to contain a section `[Permissions]` specifying the increased permissions required for specific functions in the defined firmware modules. The syntax is as follows:

```
[Permissions]
<FC> = <SFC1>:<permissions>,<SFC2>:<permissions>,...,<SFCn>:<permissions>
or for better readability
[Permissions]
<FC> = <SFC1>:<permissions>,\
    <SFC2>:<permissions>,\
    ..., \
    <SFCn>:<permissions>
```

FC is the function code (ID) of the firmware module which is exactly three hex digits with leading 0x, e.g., 0x012.

SFC is the decimal sub-function code and permissions is a hexadecimal number (maximum 8 bytes without leading 0x, e.g., FF00F000) denoting the permission in each of the eight user groups. The SFCs do not have to be ordered numerically. The permissions defined in the signed configuration file can only be used to extend the permission required, by default, for the

execution of the corresponding firmware module function. For security reasons it is not possible to suspend the required default permissions for the specified functions for external interfaces.

The signed configuration file `cmds.scf` is evaluated during startup, while all firmware modules register with their FC and SFCs at the CMDS module. In case there is an entry for a given firmware module, identified by its FC, found in the `cmds.scf` file, the required permissions for each specified SFC are set according to that definition. The list of customized permissions that are enforced after registration, is included in the audit log only if the log event "Startup messages" is activated, as by default. An entry in the Audit Log contains the following information:

```
<date> <time> <FC:0xXXX> <SFC:XX> Configured Permission=<permission>
```

For example

```
16.12.2015 13:20:45 FC:0x068 SFC:17 Configured Permission=6F000000
```

During command execution, the currently reached authentication level for the device function, identified by its FC and SFC, is compared to the required permission in the configuration file. If they are equal or no permission restrictions list exists for this module (FC), the command is executed normally performing the default (unchanged) permissions check for the command.

If the required customized permissions are not reached or in case the check of the default permissions fails, the error message B0830001 permission denied is returned.

See [Using the Signed Configuration File `cmds.scf`](#) for step-by-step instructions on how to create/edit, sign, load and activate your signed configuration file `cmds.scf`.

3.4.2.4 List of External Firmware Functions

See the `cmds_sample.cfg` file for a list of the FCs and SFCs of all external firmware functions that can be blocked, and the default permissions required for their execution, which might be increased on demand. This file is available by default in the `<install_dir>\Administration` directory.

3.5 Security Guidelines

3.5.1 General Advice

We highly recommend using strong passwords consisting of at least eight random characters, which should include uppercase and lowercase letters, special characters and random numbers.

Keep the passwords secret, do not write them down anywhere and change them regularly.

We highly recommend checking the state of the battery at regular intervals.

3.5.2 Protected Operational Environment

Before you start operating the device, ensure that the system environment is highly secure by checking that:

- No secure seal is damaged.
- PIN, PUK (personal unblocking key) or password entry cannot be monitored.
- The device is securely stored and appropriately protected against unauthorized access.
- The PIN pad is securely stored, if purchased.
- The smartcards are securely stored, if purchased.
- Only trustworthy persons have physical-/network access.
 - The administration and configuration/setup of the u.trust Anchor shall be exclusively done by verified, trusted, authorized and well-trained persons.
- Only authorized changes to the software and the configuration of device are possible.
- Regular inspections are required to deter and detect tampering (including attempts to access side-channels, or to access connections between physically separate parts of the u.trust Anchor).
- The u.trust Anchor must be protected against the possibility of attacks that are based on emanations, like electromagnetic emanations or Simple Power Analysis (SPA) or Differential Power Analysis (DPA) attacks.

Additional measures for operating an u.trust Anchor PCIe card/using an administration computer with the client application:

The following information is relevant for operating an u.trust Anchor PCIe card, and for any host PC/server where the u.trust Anchor PCIe card is integrated and where Utimaco host APIs and tools are running:

- Only trustworthy persons have a physical and network access to the u.trust Anchor and to the administration computer.
 - The administration and configuration/setup of the administration computer shall be exclusively done by verified, trusted, authorized and well-trained persons.
 - The administration computer where the u.trust Anchor PCIe card is installed in shall be placed in a highly secured area that can be only reached by authorized people. Unauthorized persons shall not have any access to the administration computer. It shall be secured by an access control mechanism, for example, password and/or smartcard.
 - The following rules apply for the passwords:
 - The minimum recommended password length is eight characters.
 - The password shall contain uppercase and lowercase letters, at least two special characters and numbers.
 - The password shall be changed periodically, at least every three months.
 - The administration computer shall be checked for malware prior to installing the u.trust Anchor card and the administration tools. Software that is not trustworthy and not required for the operation and administration of the u.trust Anchor shall be uninstalled.
 - An antivirus software with the latest updates installed shall be running on the administration computer.
 - We highly recommend using the administration computer exclusively for the operation and administration of the u.trust Anchor. There should be no Internet access and the remote computer administration should be restricted to a minimum.

3.5.3 Protection of Data Outside the u.trust Anchor

- Any externally stored data must be protected against loss, theft, unauthorized access and modification. This includes the following data:
 - For any cHSM, the MBK that is used for creating a backup of the keys of a cHSM
 - The backup copies of cryptographic keys or user backups.
 - Keys that are exported from the u.trust Anchor and keys that shall be imported into the u.trust Anchor.
 - Your Operator Base Secret (OBS).
 - The certificates of the Device Authentication Key (DAK) and any Container Authentication Key (CAK).
 - The audit data that are exported from the u.trust Anchor.
 - The private authentication keys of the users that are registered on the u.trust Anchor and are either stored in keyfiles or on smartcards.
- Any backups should be maintained in a way that ensures appropriate controls over making backups, storing backup data, and using backup data to restore an operational u.trust Anchor. The number of sets of backup data shall not exceed the minimum needed to ensure continuity of the required services.

4 Administration of u.trust Anchor with csadm

This chapter contains general information about installation requirements and installation of the csadm tool. Additionally, it can be used as a detailed reference to all commands included in csadm.

4.1 Overview of csadm

4.1.1 Syntax of csadm

The syntax of a csadm command follows this scheme:

```
csadm [Dev=...] <param1>[=...] <param2>[=...] ... <command1>[=...]
<command2>[=...] ...
```

- Parameters and commands are processed from left to right.
- If a subsequent command requires to set a parameter or to run another command prior to its own execution, it must be entered rightmost.
- Expressions in squared brackets [] are optional.
- In the syntax describing line of the individual command descriptions all parameters are shown in angle brackets < > and are explained later.
- Some parameters or commands require an assigned value (= ...), some do not.
- Some commands use a default value if none is given ([= ...]).



Certain types of shell processes treat certain characters (commas, colons, semi-colons) differently. If the execution of a csadm command fails with an error message from the shell about an illegal parameter format, quoting parameter values may be necessary.

The following is an example for a correct csadm command entry in the Microsoft PowerShell:

```
csadm [Dev=<device>] LogonSign="<user>,<keyspec>" <command>
```



The csadm tool does not support POSIX syntax. Therefore, the use of "~" is not supported. Only relative and absolute path may be specified.

Examples

```
csadm Dev=4001@192.168.1.98 LogonSign=ADMIN,d:\keys\cs2\init_dev_prv.key ...
LoadFile=c:\mdlsigalt.key
```

```
csadm Dev=4001@192.168.1.1 LogonPass=sven,ask ... AddUser=nils,00000002,hmacpwd,ask
```

Every command running on a cHSM requires the `Dev=` parameter that sets the cHSM's address. In this parameter, the slot number of the cHSM is indicated by the number contained in the form of slot 1 being `4001@`, slot 12 being `4012@` etc. cHSM slots must not be confused with MBK slots and PKCS#11 slots. Commands running locally without using a cHSM do not need this parameter.

Possible values are:

Access	Address (Dev=parameter)	Addressed Device	Description
Locally via the device node	<ul style="list-style-type: none"> Linux <code>/dev/cs2.n.m</code> Windows <code>PCI:n.m</code> <p>where n = 0 and m = {1, 2, ..., max}. The value of max depends on the product variant you use. Example values for max: 12, 31</p>	<ul style="list-style-type: none"> Linux cHSM on a u.trust Anchor PCIe card mounted in a Linux computer or in a u.trust Anchor LAN Windows cHSM on a u.trust Anchor PCIe card mounted in a Windows computer 	<ul style="list-style-type: none"> n+1 No. of local u.trust Anchor PCIe card mounted in a Linux/Windows computer or in a u.trust Anchor LAN. m No. of a cHSM (cHSM slot number) on a local u.trust Anchor PCIe card mounted in a Linux/Windows computer or in a u.trust Anchor LAN 1 = first cHSM, 2 = second cHSM etc. <p>No ports are used.</p>

Access	Address (Dev= parameter)	Addressed Device	Description
Locally via the network	4001@127.0.0.1 or 4001@localhost		Port number of a cHSM and the IP address of the Linux or the Windows computer or the u.trust Anchor LAN. Port number = basic value + cHSM slot number Typical: Port number of the first cHSM slot = 4000 + 1 = 4001, port number of the second cHSM slot = 4000 + 2 = 4002, etc.
Remotely via the network	4001@194.168.4.107 or 4001@myPC or 4001@utalan01	cHSM on a u.trust Anchor PCIe card mounted in a Linux or a Windows computer or in a u.trust Anchor LAN	The basic value 4000 is configurable by the Port parameter in the [ListenerGlad] section in the csadm.conf file (for a u.trust Anchor PCIe card mounted in a Linux or a Windows computer) or the /etc/csxlان.conf file (for a u.trust Anchor PCIe card mounted in u.trust Anchor LAN). In csadm commands, always use IP addresses without leading zeros. The host name of the computer or the u.trust Anchor LAN (using a DNS request to resolve the host name) can be used instead of an IP address.

Access	Address (Dev=parameter)	Addressed Device	Description
Locally via the network	194.168.4.107 or utalan01	u.trust Anchor LAN	IP address of the u.trust Anchor LAN (default: protocol=TCP, port=288) to access the control module of the u.trust Anchor LAN to perform <code>csadm CSL...</code> commands, for example, <code>csadm CSLGetVersion</code> . In <code>csadm</code> commands, always use IP addresses without leading zeros. The host name of the u.trust Anchor LAN (using a DNS request to resolve the host name) can be used instead of an IP address.
	127.0.0.1 or localhost		

Table 2: Possible Dev Parameters



If the environment variable `CRYPTOSERVER` is set according to the above mentioned syntax, the `Dev=` parameter can be skipped. Using the `Dev=` parameter overrides the `CRYPTOSERVER` environment variable in any case for the specific command.

4.1.2 Command Execution

The commands to the cHSM will be sent from the host to the cHSM via a TCP connection. Generally, the TCP server ('daemon') running on the device host (which is the computer directly connected to the u.trust Anchor device) forwards incoming commands to the integrated cHSM, but a few commands are responded to by the u.trust Anchor itself (e. g. setting of the TCP timeout).

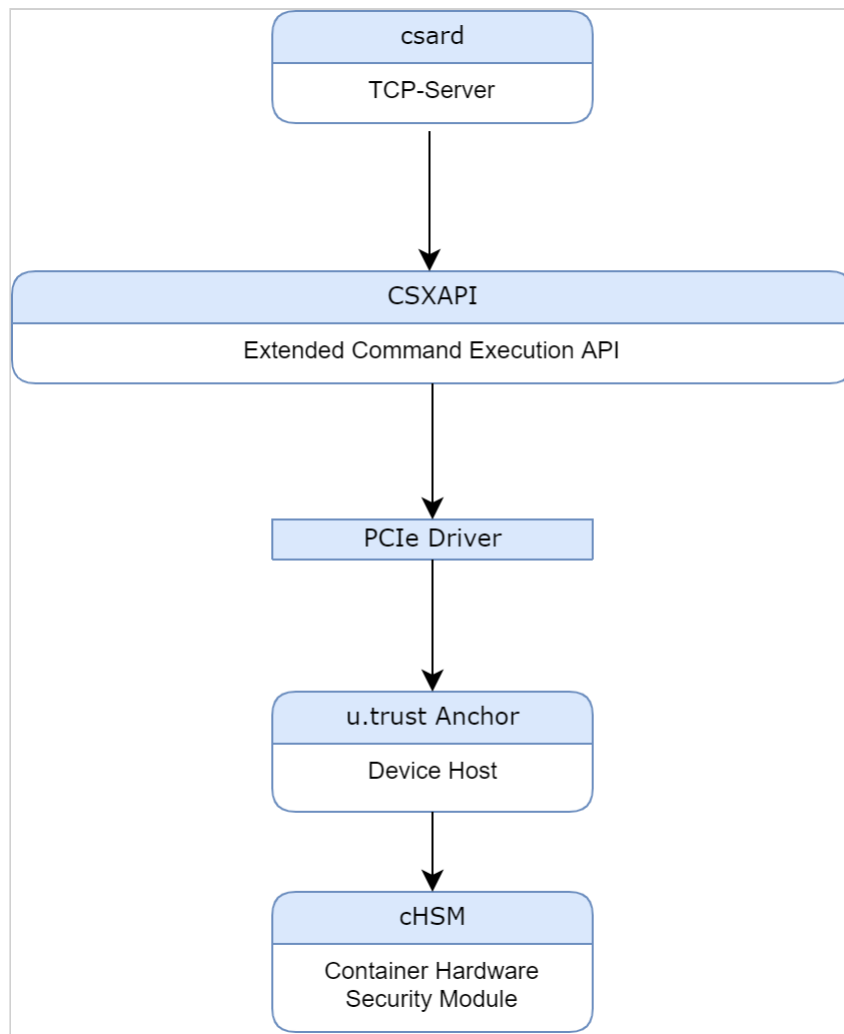


Figure 1 : u.trust Anchor csadm

An application on the host computer can use the extended command execution library CSXAPI to execute any command on a cHSM.

An application on the host computer can use the FIPS-specific library for cryptographic services (CXI library, which also contains the extended command execution library CSXAPI) to execute any of the cryptographic commands which are offered by a FIPS cHSM. CSXAPI (CryptoServer Extended Application Programming Interface) is a software running on the host which has the job to generate a C-interface out of the external cHSM byte stream

interface. (Users who are allowed to assume the role *Cryptographic User* have the right to perform these cryptographic services.)

As a standard application, the Administration Tool csadm is available to provide any kind of basic administration such as file management, setting of the cHSM's clock, user management, etc.

Commands can be divided into the following classes:

Command Destination	Counterpart on the cHSM	Command Group
cHSM	Command Scheduler Module (CMDS)	Authentication, User Management
	Administration Module (ADM)	Administration of the cHSM
	MBK Management Module (MBK)	Management of Master Backup Keys
	CXI Module (CXI)	Cryptographic services (not realized in csadm)
TCP server of the device host	Control module of the TCP Server (daemon)	Administration (configuration) of the TCP Server ('daemon')

Table 3: Command classes

4.1.3 Password Authentication

To authenticate a security-relevant administration command, operators who uses HMAC Password Authentication have to enter name and password. The password is displayed in plain text on the computer on which csadm is running.

To avoid this csadm offers hidden password entry.

For hidden password entry, enter `ask` instead of the password. csadm will then prompt for the password separately.

Example:

```
csadm LogonPass=sven,ask LoadFile=mdlsgalt.key
Enter Passphrase:
```

The password is no longer displayed in plain text but default characters are used.

The hidden password entry mechanism can also be used to hide the password of an encrypted key file or the root password from being displayed in plain text.



We strongly recommend using a hidden password entry.

4.1.4 Storage and Specification of RSA and ECDSA Keys for Authentication

Some of the csadm commands use a private or public RSA or ECDSA key to sign a command or a file, or to verify a signature. The csadm can handle these keys in three different ways:

- RSA/ECDSA keys stored in a keyfile `*.key` (as plaintext)
- RSA/ECDSA keys stored in an encrypted keyfile `*.key` (private key part stored encrypted, public key part stored as plaintext)
- RSA/ECDSA keys stored on a smartcard.

If a command needs a public key, it can be read from a file or from a smartcard. If a command needs a private key, it can either be read from a file, or a smartcard can be used to calculate the signature. In the latter case the key will not be read out of the smartcard. A PIN has to be entered via the PIN pad to enable the smartcard to generate signatures.

For security reasons, private RSA or ECDSA keys should normally be used only from smartcards or encrypted keyfiles. In a test environment, where the private key does not need to be kept secret, it may be useful to store the keys in (plaintext) files.

Encrypted RSA keyfiles are protected by a 168-bit Triple-DES key that is derived from a password with the SHA-256 hashing algorithm. Encrypted ECDSA keyfiles are protected by a 256 bit AES key that is derived from a password with the SHA-256 hashing algorithm. In both cases the password can be changed with the `csadm ChangePassword` command. With the same command a plaintext keyfile can be changed in an encrypted keyfile and vice versa (by omitting the old respectively the new password).



Apart from test environments, we strongly recommend storing private keys on smartcards. Only in this case the private key will never leave the secure token and not be used for calculations on the host.



For all commands that use RSA or ECDSA keys a key specifier is required in the command syntax. A key specifier is either a name of a keyfile (`*.key`) or a smartcard specifier.

4.1.4.1 Key Specifiers

In case that the private part of the key is needed for the command and is given in an encrypted keyfile, the password of the keyfile has to be given in the command syntax, too. This can be done by appending the password in cleartext (Example 1) directly after the filename separated by a '#' or by using a hidden password entry (Example 2):

Example 1:

```
csadm LogonSign=ADMIN,c:\my_keys\myKey.key#sIlEnCe DeleteUser=sven
```

Example 2:

```
csadm LogonSign=ADMIN,c:\my_keys\myKey.key#ask DeleteUser=sven
```

If no password is given (or the password is replaced by the string 'ask'), hidden password entry will be performed for encrypted keyfiles, which we strongly recommend.

Key Specifier Examples

Key Specifier	Description
C:\my_keys\myKey.key	Keyfile
C:\my_keys\myKey.key#mypassword	Keyfile including a password
C:\my_keys\myKey.key#ask	Keyfile including an interactive password retrieval



If RSA signature authentication with a smartcard is used and if the PIN pad is not connected to a different computer than the one csadm is running on, follow the instructions in *Connecting a PIN Pad in u.trust Anchor - Containerized Hardware Security Module (cHSM) - Administration Manual*.

For details and examples, see [LogonSign](#). See section *Authentication Mechanisms* in *u.trust Anchor - Containerized Hardware Security Module (cHSM) - Administration Manual* for details about authentication mechanisms.

4.1.4.2 Smartcard Specifiers

A smartcard specifier always starts with a colon and consists of three strings separated by colons (for example, :cs2:cjo:USB0):

- The first string identifies the type of the smartcard.
- The second string identifies the type of the smartcard reader (PIN pad).

- The last string is the name of the serial device or the USB device the reader is connected to.

Currently, only the smartcards of type TC30 and JavaCard, with the identifier cs2, are supported:

The following types of smartcard readers (PIN pads) are supported:

PIN Pad Identifier	Smartcard reader type (PIN pad types)
cyb	REINER SCT cyberJack (COM) or REINER SCT cyberJack (USB)
cjo	Utimaco cyberJack one
auto	Automatic detection of the PIN pad type


Table 4: Supported PIN pads

The following port types are supported:

Port Identifier	Description
USBn where n = {0, 1, ...}	USB port No. n+1
USBn[@<port>]@<IP address>[/<password>] [#<smartcard PIN>] where n = {0, 1, ...}	For local PIN pad and remote CryptoServer Tools/ API only

Table 5: Supported ports

In addition to that, a smartcard PIN preceded by a # can be appended. Use this option, if you do not want to enter the PIN at the PIN pad.

 The PIN you enter here is shown in plaintext in your command line. There is no option to hide it as it can be done for example for a password in the `csadm LogonSign` or the `csadm LogonPass` command.

If the PIN pad is used without using the PIN pad daemon, the smartcard PIN may even be used when a PIN is not expected. Otherwise, an error message is shown.

Example: `csadm GetCardInfo=:cs2:cjo:USB0#123456`

Key Specifier Examples

Key Specifier	Description
C:\my_keys\myKey.key	Keyfile
C:\my_keys\myKey.key#mypassword	Keyfile including a password

Key Specifier	Description
C:\my_keys\myKey.key#ask	Keyfile including an interactive password retrieval
:cs2:cjo:USBn or :cs2:auto:USBn where n = {0, 1, 2, ...}	Key from a smartcard using a Utimaco cyberJack one connected to a USB port of a Windows or Linux computer

Table 6: Examples for key specifiers

If RSA or ECDSA signature authentication with a smartcard is used and if the PIN pad is not connected to a different computer than the one csadm is running on, follow the specific instructions in *Connecting a PIN Pad in u.trust Anchor - Containerized Hardware Security Module (cHSM) - Administration Manual*.

4.1.5 Availability of commands on FIPS cHSMs

On FIPS cHSMs, some csadm commands are blocked. The following table gives an overview about all csadm commands and indicates whether these commands are available on functional FIPS cHSMs and/or FIPS error state.

Command	Available in FIPS cHSMs	Available in FIPS Error State
Help	✓	✓
PrintError	✓	✓
Version	✓	✓
Driver Commands		
Restart cHSM	✓	✓
GetInfo	✓	✓
SetTimeout	✓	✓
Administration		
GetState	✓	✓
BackupDatabase	✓	✗
RestoreDatabase	✓	✗
ListFiles	✓	✓
LoadFile	✓	✗
DeleteFile	✓	✗
GetTime	✓	✓
SetTime	✓	✗
ListFirmware	✓	✓

GetCertificates	✓	✗
LoadCertificate	✓	✗
GetBootLog	✓	✓
GetAuditLog	✓	✓
ClearAuditLog	✓	✗
GetAuditConfig	✓	✗
GenerateAuditLogKey	✓	✗
GetAuditLogKey	✓	✗
GetSignedAuditLog	✓	✗
VerifySignedAuditLog	✓	✓
ClearAuditLogFiles	✓	✗
SetAuditConfig	✓	✗
SetAdminMode	✓	✗
SetStartupMode	✓	✗
GetStartupMode	✓	✗
Test	✓	✓
User Management		
ListUser	✓	✗
AddUser	✓	✗
ChangeUser	✓	✗
Delete User	✓	✗
Backup User	✓	✗
Restore User	✓	✗
SetMaxAuthFails	✓	✗
GetMaxAuthFails	✓	✗
User Key Management		
GenKey	✓	✓
SaveKey	✓	✓
BackupKey	✓	✓
CopyBackupCard	✓	✓
GetCardInfo	✓	✓
ChangePassword	✓	✓
ChangePIN	✓	✓
Management of Master Backup Keys		
MBKListKeys	✓	✗
MBKGenerateKey	✓	✗
MBKImportKey	✓	✗

MBKCopyKey	✓	✓
MBKCardInfo	✓	✓
MBKCardCopy	✓	✓
MBKPINChange	✓	✓
Command Authentication		
LogonSign	✓	✗
LogonPass	✓	✗
ShowAuthState	✓	✗
Miscellaneous		
Cmd	✓	✓
CmdFile	✓	✓
Sleep	✓	✓

4.2 Basic Commands

These basic commands are csadm internal functions.

For their execution, no connection to the cHSM device is established.

4.2.1 Cmd

The `Cmd` command provides a generic command interface. This makes it possible to send a command as a byte stream to any firmware module without having (developed) a special tool on the administration computer.



Certain types of shell processes treat certain characters (commas, colons, semi-colons) differently. If the execution of the csadm Cmd command fails with an error message from the shell about an illegal parameter format, quoting parameter values may be necessary.

An example for the Microsoft PowerShell:

```
csadm [Dev=<device>] LogonSign=<"user">,<"keyspec">
Cmd="0x87,0x05,1,2,3"
```



Although in theory it is possible to send commands of up to 256 kB to the device using `Cmd=`, there is a restriction in praxis by the maximal command line length that can be entered if running the csadm tool on a Windows system. If a longer command byte stream shall be executed, use the `CmdFile` command..

Syntax	<code>csadm [Dev=<device>] [<Authentication>] ... Cmd=<fc>,<sfc>,<byte_n>,...</code>
Authentication	Depending on specified command
Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<fc>	Function code, module ID of the firmware module Can be either in hexadecimal format 0x00, ..., 0x3FF or in decimal format 0, ..., 1023
<sfc>	Sub function code, number of the called function Can be either in hexadecimal format 0x00, ..., 0xFF, or in decimal format 0, ..., 255
<byte_n>	n th data byte, dependent on the command Can be either in hexadecimal format 0x00, ..., 0xFF or in decimal format 0, ..., 255
Example	<code>csadm Dev=4001@192.168.1.1 LogonPass=sven,ask Cmd=0x123,0x05,1,2,3,4,5,6,7,8</code>
Output	Upon successful execution of the command, the generic command interface of the specified command is given.

4.2.2 2020-0037 CmdFile

The `CmdFile` command provides a generic command interface. Unlike the `Cmd` command it reads the input data from a file. The length of the command contained in the file may be up to 256 kBytes.

This file may contain the following characters and strings:

Character	Name	Description
#	hash sign	rest of line is comment
,	comma	separator
	space	separator
TAB	tabulator	separator
CRLF	new line	separator
hex	tag	interpret all values as hexadecimal from now on, even if '0x' is omitted
dec	tag	return to normal mode (i.e., hexadecimal interpretation requires a leading '0x')
"..."	string	string which can reach the end of the line

Table 7: Characters and strings allowed in a command file

Example

```
#
# demo command file
#
0x123  # function code / module ID
0x05   # sub function code
0x00,0,0x00,11 # hexadecimal and decimal notation may be mixed
"Hello World"  # strings are framed with quotation marks
hex 0F,1E,2D,3C,4B,5A,60,59 # leading hex-tag allows omitting of '0x'!
68,77,86,95,A4,B3,C2,D1 # still understood as hexadecimal values
dec # return to normal notation
11,22,33    # decimal values
```

Syntax

```
csadm [Dev=<device>] [<Authentication>] CmdFile=<file>
```

Authentication

Depending on specified command

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<file>	Name and extension of the file do not matter

Example

```
csadm LogonPass=sven,swordfish CmdFile=demo.cmd
```

Output	Upon successful execution of the command, the generic command interface of the specified command is given.
---------------	--

4.2.3 ConnTimeout

With this command the device connection timeout can be changed for the subsequent commands. The new timeout is valid for the current connection. Next time csadm is executed, it will use the default connection timeout setting.

Syntax	<code>csadm [Dev=<device>] ConnTimeout=<timeout> <commands></code>
---------------	--

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<timeout>	New connection timeout to be set in milliseconds
<commands>	Subsequent commands which should be executed with the new connection timeout

Example	<code>csadm Dev=4001@194.168.4.107 conntimeout=10000 GetState</code>
----------------	--

Output	Upon successful execution of the command, no output is given.
---------------	---

4.2.4 Help

If called without any parameter, this command shows a list of all available csadm commands. If a command name is given as a parameter, specific help will be provided.

Syntax	<code>csadm Help</code> <code>csadm Help=<command></code>
---------------	--

Parameter	Description
<command>	Specific csadm command to display help for

Example	<code>csadm Help=ListFiles</code>
----------------	-----------------------------------

Output	Upon successful execution, the command returns a list of all csadm command, or specific information on command and parameters if a csadm command was specified.
---------------	---

4.2.5 PrintError

This command displays the corresponding error message text to an error code. csadm has a built-in list with all standard error messages of the device and Host-APIs.

Syntax	<code>csadm PrintError=<errorcode></code>
---------------	---

Parameter	Description
<errorcode>	Error code (hexadecimal)

Example	<code>csadm PrintError=B901306F</code>
----------------	--

Output	Upon successful execution, the command returns a list of all csadm command, or specific information on command and parameters if a csadm command was specified. Error B901306F CryptoServer API LINUX can't get connection errno = 11
---------------	---

4.2.6 SetTimeout

With this command, the maximum time that the driver waits for a response from the device can be changed for the subsequent commands.



The new timeout is valid for the current connection. Next time csadm is executed, the default timeout=600000 ms will be used.

Syntax	<code>csadm [Dev=<device>] SetTimeout=<timeout> <commands></code>
---------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<timeout>	New timeout to be set in milliseconds
<commands>	Subsequent commands which should be executed with the new timeout

Example	<code>csadm Dev=4001@192.168.1.1 SetTimeout=10000 GetState</code>
----------------	---

Output	Upon successful execution of the command, the current settings of default timeout and chosen timeout are returned. default timeout: 600000 ms chosen timeout: 10000 ms
---------------	--

4.2.7 Sleep

The `Sleep` command delays the further command processing by the time given.

Syntax	<code>csadm Sleep=<time> <commands></code>
---------------	--

Parameter	Description
<time>	Time delay in seconds
<commands>	Further command whose execution should be delayed by the given time

Example	<code>csadm StartOS Sleep=3 GetState</code>
----------------	---

Output	Upon successful execution of the command, no output is given.
---------------	---

4.2.8 Version

This command shows the version numbers of the csadm tool and the included libraries.

Syntax	csadm Version
Parameter	Description
No parameters are given.	
Example	csadm Version
Output	Upon successful execution, the command returns the version of the installed csadm tool and the included libraries. csadm 2.4.2 csadm_lib (global) 3.5.0 csapi 1.11.1 (Dec 5 2018) pp_api 1.9.4 (Dec 5 2018) sl 1.1.3 yac1 1.13.0

4.2.9 StrError

This command displays the corresponding error message text to an error number (which might be system specific).

Syntax	csadm StrError=<errornumber>
Parameter	Description
<errornumber>	Error number
Example	csadm StrError=7
Output	Upon successful execution of the command, the error message text to given error number is returned.

4.2.10 RamInfo

RamInfo displays information about the current usage of the memory types SD-RAM and Secure RAM.

Syntax	csadm [Dev=<device>] RamInfo
---------------	------------------------------

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER

Example	Display memory information of the device at address 192.168.1.2 csadm Dev=192.168.1.2 RamInfo Display memory information of the device whose address is defined in the environment variable CRYPTOSERVER csadm RamInfo
----------------	---

Output	CryptoServer Internal Memory Information:				
	type	used_blocks	used_bytes	free_blocks	free_bytes
	-----	-----	-----	-----	-----
	SD-RAM	107	1079920	11	65766752
	Secure	80	28272	9	1003872

4.2.11 RamInfoCSV

RamInfoCSV displays information about the current usage of the memory type SD-RAM and Secure in CSV format. The command allows to monitor the memory usage over a longer period of time, e.g. to recognize if there are memory leaks.

Syntax	csadm [Dev=<device>] RamInfoCSV
---------------	---------------------------------

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER

Example	Display memory information of the device at address 192.168.1.2 in CSV format csadm Dev=192.168.1.2 RamInfoCSV
----------------	---

Output	<code><time>,<sd_used_blocks>,<sd_used_bytes>,<sd_free_blocks>,<sd_free_bytes>,<secure_used_blocks>,<secure_used_bytes>,<secure_free_blocks>,<secure_free_bytes></code>
---------------	---

See also `csadm RamInfo`

4.2.12 GenRandom

This command generates random data of given size.

Syntax	<code>csadm GenRandom=<size>[,<filename>]</code>
---------------	--

Parameter	Description
<code><size></code>	byte size of random data to be generated
<code><filename></code>	output filename where the generated random data will be written to; if omitted, the generated random data will be written to the standard output

Example	<pre>csadm GenRandom=64 csadm GenRandom=200,c:\test\random.txt</pre>
----------------	--

Output	<pre>c:\Utimaco\CS_4.31\Administration>csadm GenRandom=64 aa8caa7c2880361153cd7987badc8041 1d5b5803e9055ec2d087fb188f964808 0cf5eb8d867dec296158dbb3756bb63b 4491480d9b0529b9455199625ccd9a21</pre>
---------------	--

4.3 Authentication Commands

The device provides a variety of command authentication mechanisms.

Most pre-defined security-relevant commands require the authentication level 2 in a specific user group. Since each user usually has only permission 1 in one or more user groups (apart from the pre-defined user ADMIN who has permission 2 in the user groups 7 and 6, i.e., for the user management and for the administrative commands), this means that two users of the specific user group are necessary to authenticate the command. For this reason, the specific command requires authentication according to the two-person rule, i.e., authentication by two independent users is required for the command to be executed.



If a command requires multi-user authentication according to the n-person rule ($n=1\dots 16$), all n users have to apply their authentication prior to command execution:

```
csadm <Authentication_1> <Authentication_2> ... <Authentication_n> <command>
```

The users may even use different authentication mechanisms.

Example:

```
csadm LogonSign=roberta.:cs2:cjo:USB0 LogonPass=sven,ask
DeleteFile= mdlsigalt.key
```

Consider that the authentication statuses can be added without any further restriction only if the involved users are logged in to perform a command:

- of a firmware module that is not the CXI firmware module or
- of the CXI firmware module and the cryptographic key (CXI key) that is used to perform this command does not belong to any key group.

If however, the used cryptographic key in the CXI firmware module is assigned to a key group, only the authentication statuses of those logged in users can be summed up that belong to the same key group.

Example:

userA has authentication status 0x00000001 and is assigned to the key group A.

userB has authentication status 0x00000001 and is assigned to the key group B.

Three cryptographic keys are created. keyA is assigned to key group A, keyB to key group B and keyAB to key group AB.

If userA and userB are simultaneously logged in, only those commands of the CXI firmware module can be performed that only need

- an authentication status 0x00000001 for using keyA or
- an authentication status 0x00000001 for using keyB.

If a command needs

- an authentication status 0x00000001 or higher for using keyAB or
- an authentication status 0x00000002 or higher for using keyA or

- an authentication status 0x00000002 or higher for using keyB,

it cannot be performed.



The authentication commands LogonSign and LogonPass do not only perform command authentication, but provide also a Secure Messaging session for the protection of the confidentiality of the command data.

These authentication commands (leading to the necessary authentication status) can be inserted for the placeholder `<Authentication>` which are given in the syntax of all security relevant commands described in the current chapter.



Certain types of shell processes treat certain characters (commas, colons, semi-colons) differently. If the execution of a csadm command fails with an error message from the shell about an illegal parameter format, quoting parameter values may be necessary



If an authentication attempt fails, the respective user is blocked from another authentication attempt for 4 seconds.

4.3.1 LogonPass

With this command, a user with HMAC Password authentication mechanism opens an authenticated Secure Messaging session for the given command. The session key is established using the EC-Diffie-Hellman key establishment protocol.

Further authentication mechanisms can be used with this function.



To utilize mutual authentication in a Secure Messaging session, be sure to include a parameter (i.e. `OperatorRootCert=<file>`, `CustomerRootCert=<file>`, or `VendorRootCert=<file>`), otherwise there will be no verification of the respective certificate chain and hence no mutual authentication.



The authenticated Secure Messaging session will automatically close when the command execution finishes and csadm ends. For any further csadm command which needs authentication and/or confidentiality, a new authenticated Secure Messaging session has to be opened.



The usage of hidden password entry is strongly recommended. In this case, csadm requests password input (`Enter Passphrase:`) and does not echo the input.

A user uses an HMAC password-based key-derived function (HMAC-PBKDF) for authentication. The HMAC-PBKDF according to NIST SP 800-132 does not use the HMAC password itself for authentication but a function derived from the HMAC password. For HMAC-PBKDF, 1000 iterations are used here. This number of iterations, as used for the key derivation from a given password, is fix and not configurable. HMAC-PBKDF is only applied if on both sides, the host side and the firmware side, HMAC-PBKDF is applied. If it is not available on one of these sides, the legacy version of the HMAC password-based mechanism is applied, whereby the user's password is used directly as an HMAC key. In FIPS mode, using HMAC-PBKDF is mandatory.

Syntax	<code>csadm [Dev=<device>] LogonPass=<user>,<password> ... [<further Authentication>] <command></code>
---------------	--

Parameter	Description
<code><device></code>	Device specifier
<code><user></code>	User name
<code><password></code>	User password or string <code>ask</code> if hidden password entry should be used Hidden password entry is strongly recommended
<code><command></code>	Command which has to be authenticated

Example

Open an authenticated Secure Messaging session for command `DeleteFile`

- `csadm Dev=4001@192.168.1.1 LogonPass=paul,swordfish DeleteFile=mdlsigalt.key`
- `csadm LogonPass=paul,ask LogonPass=paula,ask DeleteFile=mdlsigalt.key`

Output

Upon successful execution of the command, no return is given.

Once a user with HMAC password authentication (or more precise: HMAC password-based key-derived function (HMAC-PBKDF) for authentication) has been created, this user cannot perform commands this user needs an authentication for, except for the `csadm ChangeUser` command. The `csadm ListUser` command shows the `I[1]` attribute value for this user. To enable this user to perform commands he/she needs an authentication for, he/she must change the credentials by performing the `csadm ChangeUser` command. It is important that the changes are performed by the user himself/herself and not, for example, by an administrator. Then the `csadm ListUser` command shows the `I[0]` attribute value for this user.



The parameters for the certificates need to be given **before** the `csadm LogonSign` or `csadm LogonPass` command. The `OperatorRootCert=` parameter is not relevant for u.trust Anchor Se.

4.3.2 LogonSign



For u.trust Anchor Se, only the storage location and name of the keyfile applies for the `<keyspec>` parameter. If at other locations in this documentation a smartcard specifier is used for a `csadm LogonSign` command, replace this specifier by a keyfile specifier.

With this command a user with RSA or ECDSA Signature authentication mechanism opens an authenticated Secure Messaging session for the given command. The session key is established using the EC-Diffie-Hellman key establishment protocol.

If the private part of the user's authentication key is stored on a smartcard, the user will be prompted at the PIN pad to insert their smartcard and enter the PIN. The PIN pad has to be connected to the computer where the csadm tool is running.

Further authentication mechanisms can be used with this function.



To utilize mutual authentication in a Secure Messaging session, be sure to include a parameter (i.e. `OperatorRootCert=<file>`, `CustomerRootCert=<file>`, or `VendorRootCert=<file>`), otherwise there will be no verification of the respective certificate chain and hence no mutual authentication.



The authenticated Secure Messaging session will automatically close when the command execution finishes and csadm ends. For any further csadm command which needs authentication and/or confidentiality a new authenticated Secure Messaging session has to be opened.

Syntax	<code>csadm [Dev=<device>] ... LogonSign=<user>,<keyspec> [<further Authentication>] <command></code>
---------------	---

Parameter	Description
<device>	Device specifier
<user>	User name
<keyspec>	<p>Key specifier where the private part of an RSA Signature or ECDSA Signature user's authentication key should be loaded from.</p> <p>Valid key specifiers are:</p> <ul style="list-style-type: none">▪ Smartcard specifier, e.g. <code>:cs2:cjo:USB0</code>▪ Keyfile[#password], e.g. <code>my.key#pwd</code> <p>If no password is given (or the password is <code>ask</code>, hidden password entry will be performed for encrypted key files, which is strongly recommended.</p>
<command>	Command which has to be authenticated

Example	<p>Open an authenticated Secure Messaging session for command DeleteFile</p> <pre>csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,d: \keys\prv_rsa enc.kev#ask \ DeleteFile= mdlsigalt.key</pre> <pre>csadm LogonSign=paul,:cs2:cjo:USB0 DeleteFile= mdlsigalt.key</pre> <pre>csadm LogonSign=paul,:cs2:cjo:USB0 LogonPass=paula,123456 \ DeleteFile= mdlsigalt.key</pre>
----------------	--

Output	Upon successful execution of the command, the command that was validated with it will be executed and the related output will be given.
---------------	---

4.3.3 ShowAuthState

This command displays the current authentication status on the device and a list of the users who are currently logged on to the device.

The authentication status is displayed as the sum of permissions of all users, who are currently logged on to the device.

Syntax	<code>csadm [Dev=<device>] <Authentication> ShowAuthState</code>
---------------	--

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	<ul style="list-style-type: none"> csadm Dev=4001@192.168.1.1 LogonSign=Adm1,:cs2:cjo:USB0 LogonPass=Adm2,ask ShowAuthState
----------------	--

Output	<p>Upon successful execution of the command, the current authentication state of the device is returned.</p> <pre>current AUTH state: 23000000 user: Adm1, Adm2</pre>
---------------	---

4.4 Commands for Administration

In this chapter the commands for the administration of the device are described, like status requests, alarm treatment and audit management.

Some of the commands have to be authenticated, some do not.

Command authentication is based on a user management concept, which allows the creation of various users with different permissions, authentication mechanisms and other properties.

Although command authentication is implemented in a very generic way, the way a command has to be authenticated depends on the user, i.e., on their special authentication mechanism. Therefore, the description of the command syntax does not specify each possibility of authentication. Instead, a placeholder (`<Authentication>`) is inserted, and the command example shows one possibility of authenticating the command.



Certain types of shell processes treat certain characters (commas, colons, semi-colons) differently. If the execution of a csadm command fails with an error message from the shell about an illegal parameter format, quoting parameter values may be necessary.

4.4.1 GetState

This command returns the status of the cHSM, its alarm state and setup information.



The csadm `GetState` command is responded by the device in any mode and therefore should be executed as first diagnostic measure in case of problems.

Please prepare the output of `GetState` in case of a support request.

Syntax	csadm [Dev=<device>] GetState
---------------	-------------------------------

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	csadm Dev=4001@192.168.1.1 LogonSign=Adm1,:cs2:cjo:USB0 LogonPass=Adm2,ask GetState
----------------	--

Output	<p>Upon successful execution of the command, information about the state of the device is returned.</p> <pre> mode = Operational Mode FIPS mode = ON state = INITIALIZED (0x00100004) temp = --- alarm = OFF bl_ver = 7.00.0.0 (Model: u.trust Anchor cHSM) hw_ver = 7.00.0.0 uid = 5f3dd606 b6a28c43 _= G ,I9 C adm1 = a796c345 45224b3b 893895f5 a6ddf906 EE"K; 8 adm2 = 53656375 72697479 53657276 65720000 SecurityServer adm3 = 302e3234 2e310000 00000000 00000000 0.24.1 </pre>
---------------	---

The displayed fields have the following meaning:

mode	<p>Operation mode of the device</p> <p>Operational mode The device is operational.</p> <p>Operational mode - Admin only (only available for CSAR cHSMs) The device is operational mode but with restricted command access. In this special kind of Operational Mode, all cryptographic functions are blocked, and only administration functions can be executed</p>
state	<p>State of the device</p> <p>The state will always be INITIALIZED without any alarms reported.</p> <p>For FIPS cHSMs, the FIPS error state is displayed in addition to the state INITIALIZED when FIPS self-tests fail. To leave the state ERROR, restart the u.trust Anchor cHSM.</p>
FIPS mode	<p>ON: The u.trust Anchor device is using the FIPS firmware image and cHSM is using the SecurityServer FIPS template.</p> <p>This parameter is not returned, if the u.trust Anchor device is using a non-FIPS firmware image.</p>
FIPS restrictions	<p>applied: The cHSM is using the SecurityServer FIPS template and u.trust Anchor device is using a non-FIPS firmware image.</p> <p>This parameter is not returned, if the cHSM is using a non-FIPS template.</p>
temp	Not relevant, will always display '---'
alarm	This value will always be returned as OFF , as no alarms can occur on the cHSM directly, but only on the u.trust Anchor device itself.
bl_ver	Version of the bootloader, not relevant for cHSMs
hw_ver	Version of the hardware, not relevant for cHSMs
uid	<p>UID is an 8-byte binary data field.</p> <p>The UID is a "Universal Identification" that uniquely identifies every cHSM.</p>

adm1	Cluster UUID of the cHSM, 16 bytes
adm2	Name of the template that was used to create the cHSM
adm3	Version of the template that was used to create the cHSM

Table 8: Meaning of the information fields output by the GetState command

4.4.2 SetAdminMode

This command enables the device to switch temporarily between the normal Operational Mode and the restricted Operational Mode – Administration-Only without being restarted. In Operational Mode – Administration-Only only functions needed for the device administration are available, and all cryptographic functions are blocked.

The `SetAdminMode` command can only be executed if the device is currently operating in either Operational Mode or Operational Mode – Administration-Only. It cannot be executed if the device is currently in Maintenance Mode.

The operating mode set with this command is only relevant until the next time the device is restarted. To define the restricted Operational Mode – Administration-Only mode to be active after a restart of the device, perform the `csadm SetStartupMode=1` command.

Syntax	<code>csadm [Dev=<device>] <Authentication> SetAdminMode=<mode></code>
---------------	--

Authentication	Permission 2 in the user group 6 (02000000)
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<mode>	<ul style="list-style-type: none"> 1 – Sets the device operating mode to restricted Operational Mode – Administration-Only. All cryptographic services are disabled. 0 – Sets the device operating mode back to Operational. The complete functional interface of the device can be used again.

Example	Switch the device operating mode temporarily to restricted Operational Mode – Administration-Only: csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,:cs2:cjo:USB0 SetAdminMode=1
----------------	--

Output	Upon successful execution of the command, no return is given.
---------------	---

4.4.3 SetStartupMode

With this command you can disable the automatic activation of the cryptographic interfaces of the device. The device starts, by default, in Operational Mode, which means without any restrictions on the cryptographic functions.

The `SetStartupMode` command defines the operating mode of the device – either Operational Mode or Operational Mode – Administration-Only - after a restart has been executed.

Syntax	csadm [Dev=<device>] SetStartupMode=<mode>
---------------	--

Authentication	Permission 2 in the user group 6 (02000000)
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<mode>	<ul style="list-style-type: none"> 1 – After a restart the device starts in Operational Mode – Administration-Only. All cryptographic functions are disabled. 0 – After a restart the device starts in Operational Mode, and all cryptographic functions are available again.

Example	csadm Dev=4001@192.168.1.1 LogonSign=adminUsr,:cs2:cjo:USB0 SetStartupMode=1
----------------	---

Output	Upon successful execution of the command, no output is given.
---------------	---

4.4.4 GetStartupMode

This command displays the operating mode – Operational or Operational Mode – Administration-Only - in which the device will boot after the next restart (evtl. previously set with the device command).

- If Operational Mode is displayed this implies that the full functional interface of the device will be available after a restart.
- If Operational Mode – Administration-Only is displayed this implies that the cryptographic interface of the device will be deactivated after a restart, and only administration services will be available. To make the full functional interface of the device temporarily available again, without restarting the device, use the command `SetAdminMode.`

Syntax	<code>csadm [Dev=<device>] GetStartupMode</code>
---------------	--

Parameter	Description
<code><device></code>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	<code>csadm Dev=4001@192.168.1.1 GetStartupMode</code>
----------------	--

Output	Upon successful execution of the command, the currently set startup mode of the device is returned. <ul style="list-style-type: none">▪ Operational Mode (0)▪ Operational Mode – Administration-Only (1)
---------------	---

4.4.5 GetTime

This command returns the cHSM's system time. The system clock of the cHSM has a resolution of 1/1000 second (one millisecond).

Syntax	<code>csadm [Dev=<device>] GetTime</code>
---------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	<code>csadm Dev=4001@192.168.1.1 GetTime</code>
----------------	---

Output	Upon successful execution, the command returns the date and time of the system of the cHSM in the time zone of the host PC's system time. The necessary transformation from the cHSM's internal system time (which runs in Greenwich Mean Time, GMT) to the host PC's system time zone is done automatically by csadm.
---------------	--

4.4.6 SetTime

This command sets the cHSM's system clock.

Syntax	<code>csadm [Dev=<device>] <Authentication> SetTime=<time></code>
---------------	---

Authentication	Permission 2 in the user group 6 (02000000)
-----------------------	---



The command is not sent to the cHSM until authentication is done. If this requires a lot of time, there is a gap between the given time and the actual time. If the cHSM's time has to be set very precisely, you can enter a future time and perform the last step of the authentication process when this time is reached.

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<time>	The time for the cHSM's system clock to be set to in format <code>YYYYMMDDHHMMSS</code> . The time will be converted to GMT time zone automatically.

Example	<code>csadm LogonPass=paul,swordfish SetTime=20020602115500</code>
----------------	--

Output	Any changing of the clock is taken down on the audit log file. The new entry contains the old time as well as the new time value, see section GetAuditLog . Upon successful execution of the command, no return is given. In case of an error, an error message is returned.
---------------	---

4.4.7 ListFiles

This command lists all files stored on the cHSM. The following directories are available on the different storage devices:

Device	Directory Name	Size	Description
FLASH	\FLASH	32 MBytes	Every firmware module has read and write access to the working directory. The regular set of firmware modules and any other kind of application data (databases, configuration or log files) is stored here.

Syntax	csadm [Dev=<device>] ListFiles[=<mode>]
---------------	---

Authentication	Permission 2 in the user group 6 (02000000)
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<mode>	<none> : Show all non-hidden files FLASH : All firmware modules with read and write access to the working directory will be listed. If no <code>mode</code> parameter is given, all non-hidden files are listed.

Example	csadm LogonPass=paul,swordfish Listfiles=FLASH
----------------	--

Output	<p>Upon successful execution of the command, the requested files are listed, along with their path and filename and the file size. If a directory does not contain any file, it will not be displayed.</p> <p>If the file is a firmware module, additionally the following information will be displayed:</p> <ul style="list-style-type: none"> ▪ type of files to be listed (e.g. FLASH) ▪ length of one file information block ▪ path name of the file ▪ file size ▪ abbreviation (module's short name) ▪ module ID (FC) ▪ module version ▪ extended module name ▪ CPU target type (ARM-32)
---------------	---

4.4.8 LoadFile

This command loads a file onto the working directory (`\FLASH`) of the cHSM.

In order not to violate security restrictions, only files from a whitelist are allowed to be loaded:

- Alternative Module Signature Key (file `mdlsigalt.key`)

The minimum length of an Alternative Module Signature Key is 2048 bits for u.trust Anchor FIPS.

- `*.scf` files (signed configuration files)



If the given file already exists, it will be replaced. A loaded/replaced firmware module does not become active until the cHSM has been restarted.

Syntax	<code>csadm [Dev=<device>] <Authentication> LoadFile=<file>[,<dir>]</code>
Authentication	Permission level 2 in user groups 6 and 7 (02000000)

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<file>	Filename, eventually with path If the filename contains an underscore character ('_') the rest of the name up to the file extension will be stripped before loading (e. g. <code>exmp_dbg.mtc</code> -> <code>exmp.mtc</code> , <code>adm_1.0.0.1.mtc</code> -> <code>adm.mtc</code>). The length of the file name (without directory) may not exceed 15 characters, excluding underscore characters. If no path is given with the <file> parameter, the file will be taken from the current directory. If the file to be loaded is stored on another directory, the respective path must be given.
<dir>	Directory where the file should be stored: FLASH If no directory is given, the FLASH directory is used as default.

Example	<code>csadm LogonPass=paul,swordfish LoadFile=file.scf,FLASH</code>
----------------	---



If several files should be loaded in one step, the last part of the command (`LoadFile=...`) has to be repeated for every wanted file.

Output	Upon successful execution of the command, no return is given.
---------------	---

4.4.9 DeleteFile

With this command a file or a group of files can be deleted from the cHSM's working directory (\FLASH).

In order not to violate security restrictions, only files from a whitelist are allowed to be deleted:

- Alternative Module Signature Key (file `mdlsigalt.key`)
- `*.scf` files (signed configuration files)

Syntax	<code>csadm [Dev=<device>] <Authentication> DeleteFile=[<dir>\]<file></code>
---------------	--

Authentication	Permission level 2 in user group 6 (02000000)
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<file>	Filename
<dir>	cHSM directory FLASH If no directory is given, the working directory (\FLASH) is used as default.

Example	<ul style="list-style-type: none"> ▪ csadm Dev=4001@192.168.1.1 LogonPass=paul,swordfish DeleteFile=mdlsigalt.key ▪ csadm Dev=4001@192.168.1.1 LogonSign=sm,:cs2:cjo:USB0 DeleteFile=FLASH\file.scf
----------------	---

Output	Upon successful execution of the command, no return is given.
---------------	---

4.4.10 ListFirmware

This function returns a list with information about all firmware modules which are currently running, giving their module ID, abbreviated name, CPU type of target (ARM-32 for 32-bit ARM), version number and their respective module initialization level.



If a module cannot be started or fully initialized, the Bootlog command, see [GetBootLog](#), provides more detailed information about the reason.

Syntax	csadm [Dev=<device>] ListFirmware
---------------	-----------------------------------

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	csadm Dev=4001@192.168.1.1 ListFirmware
----------------	---

Output	<p>Upon successful execution, the command returns a list of all currently running firmware modules. If for a firmware module no entry is found, the module is not loaded.</p> <p>The following information is returned:</p> <pre>ID name type version initialization level ----- 0 SMOS A32 7.1.0.0 INIT_OK 1 FIPS140 A32 7.0.0.0 INIT_OK 4 POST A32 2.2.0.0 INIT_OK a HCE A32 3.0.0.0 INIT_OK 68 CXI A32 2.4.8.1 INIT_OK 81 VDES A32 2.2.0.0 INIT_OK 83 CMDS A32 3.8.1.0 INIT_OK 84 VRSA A32 2.2.0.0 INIT_OK 86 UTIL A32 3.0.7.1 INIT_OK 87 ADM A32 3.1.2.0 INIT_OK 88 DB A32 2.0.0.2 INIT_OK 89 HASH A32 2.2.0.0 INIT_OK 8b AES A32 2.2.0.0 INIT_OK 8d DSA A32 2.2.0.0 INIT_OK 8e LNA A32 2.2.0.0 INIT_OK 8f ECA A32 2.2.0.0 INIT_OK 91 ASN1 A32 2.2.0.0 INIT_OK 96 MBK A32 2.5.1.1 INIT_OK 9c ECDSA A32 2.2.0.0 INIT_OK 9f CRYPT A32 2.2.0.0 INIT_OK</pre>
---------------	---

Parameter	Description
ID	The ID of the firmware module
type	The CPU type of the firmware module ARM-32: 32-bit ARM
version	The version of the firmware module

Parameter	Description
initialization level	INIT_NONE The module is present but the initialization of the module has not been started yet. This entry will be made by the OS when loading the module into the DDR2 RAM.
	INIT_INTERNAL The module has finished its internal initialization. "Internal" means all initialization tasks that can be done without using services from other modules like memory allocation, FIFO creation, global data initialization, etc.
	INIT_DEP_OK The module has successfully completed the check of dependencies on other modules (second step of initialization). "Dependencies on other modules" means that the module is dependent on services provided by other modules in order to run correctly.
	INIT_OK This is the highest possible level: The initialization of the module is completed (third step of initialization). Possible calls to services from other modules are done successfully.
	INIT_FAILED Module initialization failed. Services provided by this module are not available
	INACTIVE Module is loaded but cannot supply services, e.g. because of not-installed hardware components. This initialization level is currently only used by firmware module HCE.
	SUSPENDED Module was shutdown and cannot supply services any more

Table 9: Meaning of the csadm ListFirmware return parameters

4.4.11 GetBootLog

`GetBootLog` returns the boot log file, which contains log messages that are made during the boot process. The log messages are made by the operating system and other firmware modules or by the bootloader if the command is called in Bootloader Mode. The boot log is held in working memory and is not written into a permanent file. In this way, the content of the previous boot log file is cleared every time the operating system starts.

Syntax	<code>csadm [Dev=<device>] GetBootLog</code>
---------------	--

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	csadm Dev=4001@192.168.1.1 GetBootLog
Output	<p>Upon successful execution, the command returns the boot log.</p> <pre> --....-- --:--:-- SMOS Ver. 7.1.0.0 (Jan 1 1970) started [0] --....-- --:--:-- Compiler Ver. 9.3.0 --....-- --:--:-- CPU clock frequency: 1000000000 --....-- --:--:-- trng_init --....-- --:--:-- Sensory Controller Ver. 2.0.0.42 [0/0] --....-- --:--:-- Real Random Number Generator initialized with: RESEED_INTERVAL = 1000 PREDICTION_RESISTANCE = 0 REALRANDOM_SHARE = 3 --....-- --:--:-- Pseudo Random Number Generator initialized with: RESEED_INTERVAL = 1000 PREDICTION_RESISTANCE = 0 REALRANDOM_SHARE = 0 --....-- --:--:-- Load module 'fips140.msc' from FLASHFILE --....-- --:--:-- FIPS module has no list of FIPS approved modules! --....-- --:--:-- FIPS strict mode --....-- --:--:-- Signed Licence File found: dev.slf --....-- --:--:-- Load module 'adm.msc' from FLASHFILE --....-- --:--:-- Load module 'cmds.msc' from FLASHFILE --....-- --:--:-- CMDS: .pscf support enabled --....-- --:--:-- Load module 'crypt.msc' from FLASHFILE --....-- --:--:-- Load module 'cxi.msc' from FLASHFILE --....-- --:--:-- Load module 'db.msc' from FLASHFILE --....-- --:--:-- Load module 'hce.msc' from FLASHFILE --....-- --:--:-- Load module 'mbk.msc' from FLASHFILE --....-- --:--:-- Load module 'ntp.msc' from FLASHFILE --....-- --:--:-- Load module 'util.msc' from FLASHFILE --....-- --:--:-- module 0x01 (FIPS140) initialized successfully --....-- --:--:-- module 0x83 (CMDS) initialized successfully --....-- --:--:-- module 0x86 (UTIL) initialized successfully --....-- --:--:-- CSAR XRS Accelerator detected --....-- --:--:-- module 0x0a (HCE) initialized successfully --....-- --:--:-- module 0x9f (CRYPT) initialized successfully --....-- --:--:-- module 0x88 (DB) initialized successfully --....-- --:--:-- MBK: Cannot auto generate key (B0001100) --....-- --:--:-- module 0x96 (MBK) initialized successfully --....-- --:--:-- module 0x68 (CXI) initialized successfully --....-- --:--:-- CMDS: .pscf support enabled --....-- --:--:-- module 0x87 (ADM) initialized successfully --....-- --:--:-- CMDS: all certificates for authentication successfully loaded. --....-- --:--:-- CMDS: successfully initialized in FIPS mode </pre>

4.4.12 GetAuditLog

The `csadm GetAuditLog` command shows the contents of all audit log files. This command can be performed in any mode and does not need any authentication.

Syntax	<code>csadm [Dev=<device>] GetAuditLog</code>
---------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	<code>csadm Dev=4001@192.168.1.1 GetAuditLog</code>
----------------	---

Output	Upon successful execution, the command returns the audit log entries. <pre> ___.___.__ __:__:__ [] FC:0x000 SFC: SMOS Ver. 7.1.0.0 successfully started ___.___.__ __:__:__ FC:0x087 SFC:0x07 Set Time [b087000f]</pre>
---------------	--

4.4.13 ClearAuditLog

This function erases the content of audit logfiles. For a continuous auditing the newest logfile(s) can be kept.

Consider that there is another csadm command for deleting audit log files, `csadm ClearAuditLogFiles`. However, that command can be applied to audit log files with long file names only (for example, audit00003A.log).

You can verify which audit log files are present in the FLASH memory of the CryptoServer by performing a command according to the following example:

```
csadm Dev=4001@192.168.1.1 ListFiles | grep .log
```

Syntax	<code>csadm [Dev=<device>] <Authentication> ClearAuditLog[=<n>]</code>
---------------	--

Authentication	Permission level 2 in user group 6 or 7 (e.g. 02000000).
-----------------------	--

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<n>	n newest logfiles will not be deleted

Example	csadm LogonSign=ADMIN,:cs2:cjo:USB0 ClearAuditLog=2
----------------	---

Output	Upon successful execution of the command, no output is given.
---------------	---

4.4.14 GetAuditConfig

This command shows the configuration setting for the audit functionality of the device.

Syntax	csadm [Dev=<device>] GetAuditConfig
---------------	-------------------------------------

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	csadm GetAuditConfig
----------------	----------------------

The audit log configuration parameters have the following meaning.

4.4.15 Test

With this command, a communication test (echo loopback test) with the device is executed. For each loop it sends random test patterns to the device, beginning with the minimum data length, which will be incremented until the maximum data length is reached. The device echoes the received command. On the host side, the received answer is compared with the command originally sent.

Syntax	<pre>csadm [Dev=<device>] Test=[<datalength>,<loopcount>] csadm [Dev=<device>] Test=<min_datalength>,<max_datalength>[,<increment>],<loopcount></pre>
---------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSEVER.
<datalength>	Data length [bytes] of the used pattern. <ul style="list-style-type: none"> if a value is set: <pre>min_datalength = max_datalength = value</pre> if omitted: <pre>min_datalength = 0, max_datalength = 131072 (128kB)</pre>
<min_datalength>	Minimum data length in bytes of the used pattern
<max_datalength>	Maximum data length in bytes of the used pattern (max: 131072)
<increment>	Data length increment [bytes]; default = 1
<loopcount>	Number of execution cycles

Example	<ul style="list-style-type: none"> csadm Test=256,10000 csadm Test=0,128,2,10
----------------	---



A little time is needed to create the test patterns for each loop. A pure benchmark test leads to slightly better results.

Output	<p>Upon successful execution of the command, the result data of the executed test command is given.</p> <pre>Data type : random data length (min): 256 data length (max): 256 increment : 1 loop count : 1000 len count 256 1000 execution time : 480 ms data throughput : 533333 bytes/s transaction time : 0.480000 ms</pre>
---------------	--

4.4.16 SetAuditConfig

With this command the configuration of the audit functionality of the device can be changed.

Syntax	<pre>csadm [Dev=<device>] <Authentication> ... SetAuditConfig=<param1=value1>[,<param2=value2>,<param3=value3>, ...]</pre>
---------------	--

Authentication	Permission level 2 in user group 7 and level 2 in user group 6 (22000000)
-----------------------	---



If a configuration parameter is not set, it remains unchanged.

Example	<ul style="list-style-type: none"> csadm LogonPass=svenpaul,swordfish ... SetAuditConfig=MaxFileCount=5,Events=1:2:3 csadm LogonSign=nils,:cs2:cjo:USB0 ... SetAuditConfig=MaxFileCount=3,MaxFileSize=200000,Events=0x0000007
----------------	---

Output	Upon successful execution of the command, no output is given.
---------------	---

Class	Meaning
1	Firmware management (default)

Class	Meaning
2	User management (default)
3	Date/Time management (default)
4	Startup messages (default)
5	Audit log management (default)
6	MBK management (default)
7	Key management
8	Successful login
9	Failed login (default)
10	Backup/Restore (default)
11	Operating system events (default)
12	Action needed (default)
13-24	Future use
25-31	Customer definition

Table 10: Audit Message Classes

4.4.17 GetCertificates

This command initiates a Secure Messaging handshake with the cHSM. It saves the certificates sent by the device to the given directory.

The certificates will be DER-encoded.

Syntax	<code>csadm [Dev=<device>] GetCertificate[=<dir>[,<flags>]]</code>
---------------	--

Authentication	Permission level 2 in user group group 6 (02000000)
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<dir>	Certificates will be written to this directory Default: working directory
<flags>	Optional flags overwrite : Certificate files will be overwritten if already existing

Example	<code>csadm Dev=4001@192.168.1.1 GetCertificates=/tmp/hsm_certs/</code>
----------------	---

Output

Upon successful execution of the command, no output is given.

4.4.18 LoadCertificate

This command loads a certificate for the Device Authentication Key Certificate signed by your own CA. This can be used as an additional authentication mechanism on establishing secure messaging. An already loaded certificate is overwritten by the new certificate if it is in the right format. Currently, only X509v3 certificates in `.der` format are accepted.

Syntax

```
csadm [Dev=<device>] LoadCertificate
```

Authentication

Permission level 2 in user group 6 (02000000)

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<file>	File name and possibly the path of the file holding the certificate

Example

```
csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,d:\keys\init_prv.key \  
LoadCertificate=hsm:certificate.der
```

Output

Upon successful execution of the command, no output is given.

4.4.19 GenerateAuditLogKey

This command generates and prints out an asymmetric key as Audit Log Signature Key which will be stored in the database `auditkey.db`. It can only be performed if the `csadm GenerateAuditLogKey` command has been performed on the device before.

When this command is performed, the private part of the Audit Log Key is saved within the `auditkey.db` database on the device and the public part of the Audit Log Key is returned as the output of the command.

If the database already exists, an error message is returned.

Syntax	<code>csadm [Dev=<device>] <Authentication> GenerateAuditLogKey=<mode></code>
Authentication	Permission level 2 in user group 6 (02000000)
Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<mode>	1: 3072 bit RSA key with public exponent 0x10001, PKCS#1 RSA PSS signature with SHA-256 and 32 bytes salt length 2: EDSA with NIST P-256 curve and SHA-256
Example	<code>csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,:cs2:cjo:USB0 GenerateAuditLogKey=1 > ./my_key_files.txt</code>
Output	Upon successful execution, the command returns the AuditLogKey.

4.4.20 GetAuditLogKey

This command retrieves and prints an existing asymmetric key from the database and outputs its public part. Performing the command does not create any audit log entry.

Syntax	<code>csadm [Dev=<device>] GetAuditLogKey</code>
Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
Example	<code>csadm Dev=4001@192.168.1.1 GetAuditLogKey</code>
Output	Upon successful execution, the command returns the existing asymmetric audit log key from the database.

4.4.21 GetInfo

This command retrieves information from the PCIe driver of the u.trust Anchor device. It shows the driver version, PCIe slot number, interrupt number, and the state of the error correction.

The command is used for diagnostic purposes in case of an error.



The `GetInfo` command will be executed even if the cHSM does not respond to any command (even the `GetState` command).

Please prepare this output in case of a support request.

Syntax

```
csadm [Dev=<device>] GetInfo
```

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example

```
csadm Dev=4001@192.168.1.1 GetInfo
```

Output

Upon successful execution, the function returns several parameters that are relevant in case of a support request. Not all parameters will be explained here, but some of the parameters below can give you useful information.

Parameter	Description
driver version	Version of the u.trust Anchor's PCI/PCIe driver installed on the host system <code><major>.<minor>.<patch></code>
LogLevel	Level of the log entries of the driver in the kernel log 0 = none 1 = error 2 = warning 3 = info (default) 4 = trace 5 = debug
Device	Name of device node in <code>/dev</code>

<i>Parameter</i>	<i>Description</i>
slot	PCIe slot number of the host computer where the u.trust Anchor PCIe card is mounted
devfn	Device function pf = physical function vf = virtual function
chnid	channel ID, 'recipient address' or 'cHSM Id'
state	offline disabled initialized enabled
sriov_mode	Function, (device,channel), can be used for virtualization (SRIOV)
relaunch	Counts how often the device was restarted
HW version	hardware version 7.3.xx
FPGA version	7.3.x.x

Table 11: Meaning of output parameters of csadm GetInfo

4.4.22 GetBattState

The `csadm GetBattState` command is only performed on the u.trust Anchor PCIe card. It delivers the state of batteries related to this u.trust Anchor PCIe card.

This command shows the state of the two device batteries – the Carrier Battery and the External Battery. The state 'low' indicates that the battery should be renewed as soon as possible to avoid a power low alarm.



The External Battery is only relevant for a LAN device. For a PCIe card the External Battery is not present, and thus the state 'absence' for the External Battery is no reason to worry.

Syntax	<code>csadm [Dev=<device>] GetBattState</code>
---------------	--

<i>Parameter</i>	<i>Description</i>
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	<pre>csadm Dev=4000@192.168.1.1 GetBattState or csadm Dev=/dev/cs2.0 GetBattState</pre>
Output	<p>Upon successful execution of the command, the current state of the battery is returned.</p> <pre>Carrier Battery: ok (3.055 V) External Battery: absence</pre>

4.4.23 GetSignedAuditLog

This command retrieves all signed audit log files. Each signed audit log file will be named `<serial_number>_<file_number>.log` and stored in a given output directory. The serial number is the UID of the cHSM, the file number is formatted as 8 hexadecimal digits.

Syntax	<pre>csadm [Dev=<device>] <Authentication> [AuditPubKey=<keyfile>] GetSignedAuditLog=<outdir></pre>
---------------	---

Authentication	Permission level 1 in user group 6 (01000000)
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<keyfile>	The path to the audit log signature key file containing the public part of the audit log signature key. This value can be omitted if the path to this file is set as the environment variable CS_AUDIT_KEYS.
<outdir>	The path of the output directory where the signed audit log files will be stored. The content of the log files is not encrypted but signed. Each signed audit log file is named according to the schema <code><serial_number>_<file_number>.log</code> and stored in the specified output directory. <code><serial_number></code> is the serial number of the device and <code><file_number></code> is an 8-digit hexadecimal number without a leading 0x. Example name of an audit file on the host: <code>CS123456_0013AC97.log</code>

Example	<pre>csadm Dev=4001@192.168.1.1 AuditPubKey=myauditkey.txt GetSignedAuditLog=logs</pre>
----------------	---

Output	Upon successful execution, no output is given.
---------------	--

4.4.24 VerifySignedAuditLog

This command verifies the given signed audit log files. Performing this command does not create any audit log entry.

Syntax	<code>csadm [AuditPubKey=<keyfile>] VerifySignedAuditLogfile=<file>[,print]</code>
---------------	--

Parameter	Description
<keyfile>	The path to the audit log signature key file containing the public part of the audit log signature key, to be retrieved as described in sections GenerateAuditLogKey and GetAuditLogKey . This value can be omitted if the path to this file is set an environment variable <code>CS_AUDIT_KEYS</code> .
<file>	The signed audit log file to be verified This file has to be retrieved previously as described in section GetSignedAuditLog .
<print>	If this flag is given, the contents will be printed after successful verification. If this flag is not given, the text <code>OK</code> will be printed in case of successful verification.

Example	<code>csadm AuditPubKey=myauditkey.txt VerifySignedAuditLogfile=CS123456_0013AC97.log,print</code>
----------------	--

Output	Upon successful execution of the command, the contents of the signed audit log files will be printed if the parameter <code>print</code> was given. Otherwise, the success message <code>OK</code> will be returned.
---------------	--

4.4.25 ClearAuditLogFiles

This command clears all audit log files up to and including file number `<n>`. If the parameter `<outdir>` is given, the file(s) will only be deleted if the file `<n>` stored on the host has the same content as the file `<n>` on the device.

It does not delete signed audit log files with extra long file names that have been retrieved and stored on the computer (host) csadm is running on by performing the `csadm GetSignedAuditLog` command.

Consider that there is another csadm command for deleting audit log files, `csadm ClearAuditLog`. That command can be applied to audit log files with short file names (for example, `audit_00.log`) and long file names (for example, `audit00003A.log`).

You can verify which audit log files are present in the FLASH memory of the device by performing a command according to the following example:

```
csadm Dev=4001@192.168.1.1 ListFiles | grep .log
```

Syntax	<code>csadm [Dev=<device>] <Authentication> ClearAuditLogFiles=<n>[,outdir]</code>
---------------	--

Authentication	Permission level 2 in user group 6 or 7 (02000000)
-----------------------	--

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<n>	The file number in hexadecimal notation (without leading 0x) Alternatively, the string <code>all</code> can be given, in which case all files will be deleted.
<outdir>	The path to the output directory in which the signed audit log files are stored. If the <outdir> directory is specified, it is verified whether the contents of the audit log files in the FLASH memory on the device specified by <n> is identical to the corresponding audit log files in the <outdir> directory on the host. If this is the case, the specified audit log files on the device are deleted. If one or more files are missing or do not match, no file is deleted. The names of the audit log files on the device differ in any case from the names of the corresponding files on the host (long file names vs. extra long file names). The <code>all</code> value cannot be combined with the <outdir> parameter.

Example	Delete all audit log files up to and including number 00003a: <code>csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,:cs2:cjo:USB0 ClearAuditLogFiles=3a</code>
----------------	--

Output	Upon successful execution of the command, no output is given.
---------------	---

4.4.26 LoadAltMdlSigKey

This command loads the public part of the customer's Alternative Module Signature Key into the device. This key is later used to verify the authenticity of self-programmed firmware

modules during the load process, which are signed by the customer with the private part of his Alternative Module Signature Key.

The minimum length of an Alternative Module Signature Key is 2048 bit for u.trust Anchor FIPS.

Syntax	<code>csadm [Dev=<device>] <Authentication> LoadAltMdlSigKey=<keyspec></code>
---------------	---

Authentication	Permission level 2 in user group 6 and 7 (22000000)
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<keyspec>	Key specifier of the public part of the customer's Alternative Module Signature Key

Example	<code>csadm LogonSign=ADMIN,:cs2:cjo:USB0 LoadAltMdlSigKey=MyMdlSig.key</code>
----------------	--

Output	Upon successful execution of the command, no output is given.
---------------	---



The key will be stored on the device as file `mdlsigalt.key`. If the key already exists, it will be overwritten.



Firmware modules created by Utimaco IS GmbH are always signed with the Utimaco Module Signature Key. Since the public part of this key is loaded into every device the operating software is always able to verify the signature on load of any firmware module which is created by Utimaco IS GmbH.

4.4.27 BackupDatabase

This command creates a backup of all entries of the given device database (for example, the cryptographic key database (`CXIKEY.db`) or the audit log signature key database (`auditkey.db`)).

The function creates a file with the name of the database and stores it in the current directory of the command line. Each database entry in the created backup file is encrypted with the Master Backup Key (MBK) of the device. Due to a check value (MAC calculated with the MBK) over each database record it is not possible to change any item (for example, database index, key record).

The database for which the backup shall be created must have the file extension `*.db` . The function creates a file with ASCII characters and additional information about the used MBK. If a file with the name of the database already exists in the current directory, it will be overwritten. It is not possible to backup the MBK database or the database with the secure messaging session keys.



Perform the `csadm MBKListKeys` command to determine which Master Backup Key (MBK) is currently in use in MBK slot 3 by the device. This MBK is used by the `csadm BackupDatabase` command to protect the backup file to be generated. If the MBK in MBK slot 3 is the autogenerated MBK named `AUTO-GEN` , the `csadm BackupDatabase` command cannot be performed. Import a different MBK into MBK slot 3 using the `csadm MBKImportKey` command.

It is important to note down which MBK has been used because for a successful restoring of this backup file at a later date it is necessary that the same MBK is in MBK slot 3. Otherwise, for example, after the execution of a `csadm MBKImportKey` command or after an MBK rollover, the backup file is inaccessible.



Only one database backup can be saved at a time. To back up several databases, the command must be executed separately for each individual database.

As of SecurityServer 6.0.0, a new database backup structure/format has been introduced.

- For creating and restoring backups, it is recommended to use a csadm version that matches the firmware version.

- Backups with the format earlier than 6.0.0 are still possible to be restored, except in FIPS mode because FIPS does not allow the old format.
- Restoring a backup with the new format into firmware version < 6.0.0 is not possible. Generally, restoring backups of a newer firmware version into older firmware is not supported.
- As of version 6.0.0, customers must use the `csadm BackupUser` command to back up users, because `csadm ... BackupDatabase=user.db` is not supported anymore.



Certain types of shell processes treat certain characters (commas, colons, semi-colons) differently. If the execution of a csadm command fails with an error message from the shell about an illegal parameter format, quoting parameter values may be necessary.

The following is an example for a correct csadm command entry in the Microsoft PowerShell:

```
csadm [Dev=<device>] LogonSign="<user>,<keyspec>" <command>
```

Syntax

```
csadm [Dev=<device>] <Authentication> BackupDatabase=<name of database>
```

Authentication

Permission level 2 in user group 6 and 7 (22000000)

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<name of database>	Name of the device database

Example

Back up the cryptographic key database

```
csadm LogonSign=ADMIN,:cs2:cjo:USB0  
BackupDatabase=CXIKEY.db
```

Output

Upon successful execution of the command, no output is given.

4.4.28 RestoreDatabase

This command restores a database on the device from a backup file that was created with the `csadm BackupDatabase` command.

For the restore procedure, the function takes each record from the backup file, decrypts it inside the device with the Master Backup Key (MBK), verifies the MAC and stores it under the given database index. If an entry with the given database index already exists, it will be overwritten. The backup file must have the file extension `*.db`. It is not possible to restore the MBK database or the database with the secure messaging session keys.

As of SecurityServer 6.0.0, a new database backup structure/format has been introduced.

- For creating and restoring backups, it is recommended to use a csadm version that matches the firmware version.
- Backups with the format earlier than 6.0.0 are still possible to be restored, except in FIPS mode because FIPS does not allow the old format.
- Restoring a backup with the new format into firmware version < 6.0.0 is not possible. Generally, restoring backups of a newer firmware version into older firmware is not supported.
- As of version 6.0.0, customers must use the `csadm BackupUser` command to back up users, because `csadm ... BackupDatabase=user.db` is not supported anymore.



As of SecurityServer FIPS 140-3 6.0.0, a new firmware and a new backup format are applied. As a consequence, if you have backed up a database using a SecurityServer < 6.0.0 (i.e. using the old firmware with the old backup format), this database backup cannot be restored in one step by SecurityServer FIPS 140-3 >= 6.0.0.

Instead, proceed as follows:

1. Restore the database backup using SecurityServer non-FIPS >= 6.0.0.
2. Create a new database backup using SecurityServer non-FIPS >= 6.0.0 (i.e. using the new firmware with the new backup format).
3. Restore the new database backup using SecurityServer FIPS 140-3 >= 6.0.0 (i.e. using the new firmware with the new backup format).



We recommend performing a csadm Restart after executing the `RestoreDatabase` command:

If CXI configuration items are stored in the restored database, they are applied immediately to the CryptoServer. Other restored configuration items will not become effective until a CryptoServer restart will have been performed.



The same Master Backup Key (MBK) which has been used to create the backup file has to be stored on the target device.



Certain types of shell processes treat certain characters (commas, colons, semi-colons) differently. If the execution of a csadm command fails with an error message from the shell about an illegal parameter format, quoting parameter values may be necessary.

The following is an example for a correct csadm command entry in the Microsoft PowerShell:

```
csadm [Dev=<device>] LogonSign="<user>,<keyspec>" <command>
```

Syntax

```
csadm [Dev=<device>] <Authentication> ...  
RestoreDatabase=<backup_file>
```

Authentication

Permission level 2 in user group 6 and 7 (22000000)

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<backup_file>	Name of the backup database file (eventually with path)

Example

```
Restore the cryptographic key database  
csadm LogonSign=ADMIN,:cs2:cjo:USB0  
RestoreDatabase=CXIKEY.db
```

Output

Upon successful execution of the command, no output is given.

4.4.29 Restart

This command performs a reboot of the cHSM.

Syntax	<code>csadm [Dev=<device>] Restart</code>
---------------	---

Parameter	Description
<code><device></code>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	<code>csadm Dev=4001@192.168.1.1 Restart</code>
----------------	---

Output	Upon successful execution of the command, the cHSM restarts.
---------------	--

4.4.30 SignConfig

The `csadm SignConfig` command signs a self-created configuration file `cmds.cfg` (or any other file extension) with the Alternative Module Signature Key. The signed configuration file is specified by the file extension `*.scf`, and allows you to define specific firmware module configuration settings, for example, to disable some selected device functions. The syntax of the signed configuration file follows the syntax of configuration files.

The minimum length of an Alternative Module Signature Key is 2048 bit for u.trust Anchor FIPS.

Syntax	<code>csadm MdlSignKey=<keyspec> SignConfig=<file></code> If the Module Signature Key is stored in a device: <code>csadm [Dev=<device>] <Authentication> MdlSignKey=<keyspec> SignConfig=<file></code>
---------------	--

Authentication	Permission level 2 in user group 6 and 7 (22000000)
-----------------------	---

Parameter	Description
<keyspec>	<p>Key specifier for the MdlSignKey which is the private part of the Module Signature Key or the Alternative Module Signature Key. Can be defined as:</p> <ul style="list-style-type: none"> Smartcard specifier If the Module Signature Key or the Alternative Module Signature Key is stored on a smartcard, for example, <code>:cs2:cjo:USB0</code> Keyfile specifier (keyfile name and, if needed, file path) If the Module Signature Key or the Alternative Module Signature Key is stored in a keyfile In case an encrypted keyfile is used, the correct password for the keyfile is required, i.e., <code><keyfile>#password</code> . <code>password</code> is needed in case of an encrypted keyfile. If no password is given (or the password is replaced by the string 'ask'), hidden password entry will be performed for encrypted keyfiles, which we strongly recommend. If the Module Signature Key is stored in the cryptographic key database, <code>CXIKY.db</code> , of a (LAN) device: <code>Dev:<keyname>[,<key_spec(version)>]</code> , for example, <code>Dev:CS_MSK,1</code> <code>Dev:</code> references the device address defined in the parameter <code>Dev=<device></code> or in the environment variable CRYPTOSERVER. Do not change this string. The keyname is the name of the keyfile without file extension *.key. It should be followed by the key specifier <code>key_spec(version)</code> , if any was defined on key generation.
<file>	Name and storage location of the configuration file (<code>cmds.*</code>) to be signed

Example	<p>Sign a configuration file by using a Module Signature Key stored in a keyfile. <code>csadm MdlSignKey=D:\keys\myMDLSign.key SignConfig=C:\myConfig\cmds.cfg</code></p> <p>Sign a configuration file by using a Module Signature Key stored on a smartcard. <code>csadm MdlSignKey=:cs2:cjo:USB0 SignConfig=C:\myConfig\cmds.txt</code></p> <p>Sign a configuration file by using a Module Signature Key stored in a CryptoServer LAN device. <code>csadm Dev=192.168.0.1 LogoSign=SysAdmin,:cs2:cjo:USB0 MdlSignKey=Dev:CS_MSK,1 SignConfig=C:\myConfig\cmds.txt</code></p>
----------------	---

Output	Upon successful execution of the command, no output is given.
---------------	---

Please find detailed information on how to use a signed configuration file to disable selected device functions or to increase the authentication requirements for selected device functions in [Using the Signed Configuration File cmds.scf](#).

4.5 User Management Commands

The device provides a sophisticated user management in order to maintain different users with different authentication mechanisms and permissions.

User's credentials are stored in the user database `user.db`, which is hosted on the device.

The respective commands to create, change or delete users are described in this chapter. Within this user management, the following properties can be assigned to each user:

<i>Property</i>	<i>Description</i>
Name	Username, up to 255 printable characters (must not contain a '~')

Property	Description																					
Permission	<p>A user's permission, for example, 2A000000, is an eight-digit sequence of hexadecimal values between 0 and F (15). Each digit stands for one of the eight device specific user groups. Each user group is assigned a specific group of functions/commands, and is uniquely identified by a number in the range from 0 to 7. User's permission in the user group 7 is displayed as the most left one, and the permission in user group 0 as the most right one. This means, user's permission in a specific user group defines the rights a user is granted on creation, allowing him to execute security-relevant functions on the device that always require user authentication.</p> <p>The value 0 means that the user has no rights in that particular user group.</p> <ul style="list-style-type: none">▪ 1 means two-person rule: although the user can authenticate commands to the device, a second user is also required to do this.▪ 2 means that the user is entitled to authenticate commands on his own. <p>Values 3 to F mean that the user can authenticate commands on his own as well as can participate in performing the n-person rule authentication (n > 2), and M-of-N users authentication (where N is the number of all device users with the same role-based user profile, M is the number of users with the same role-based user profile required to authenticate specific command/function). 2 is the highest possible permission required by the device functions/commands by default. 3 to F might be required by functions with custom permissions defined in a signed configuration file <code>cmds.scf</code>.</p> <p>Some user groups are reserved by Utimaco for applications and their corresponding role-based user profiles as shown in the next table.</p>																					
	<table><tr><th>User Group</th><th>Application</th><th>Role-based user profile and permissions</th></tr><tr><td>0</td><td>All cryptographic interfaces</td><td>Cryptographic User und PKCS#11 User; 00000002</td></tr><tr><td>1</td><td>PKCS#11</td><td>PKCS#11 Key Manager; 00000020</td></tr><tr><td>2</td><td>PKCS#11</td><td>PKCS#11 Security Officer (SO); 00000200</td></tr><tr><td>5</td><td>NTP Administration</td><td>NTP Manager; 00200000</td></tr><tr><td>6</td><td>System Administration</td><td>System Manager; 02000000</td></tr><tr><td>7</td><td>User Management</td><td>User Manager; 20000000</td></tr></table>	User Group	Application	Role-based user profile and permissions	0	All cryptographic interfaces	Cryptographic User und PKCS#11 User; 00000002	1	PKCS#11	PKCS#11 Key Manager; 00000020	2	PKCS#11	PKCS#11 Security Officer (SO); 00000200	5	NTP Administration	NTP Manager; 00200000	6	System Administration	System Manager; 02000000	7	User Management	User Manager; 20000000
	User Group	Application	Role-based user profile and permissions																			
	0	All cryptographic interfaces	Cryptographic User und PKCS#11 User; 00000002																			
1	PKCS#11	PKCS#11 Key Manager; 00000020																				
2	PKCS#11	PKCS#11 Security Officer (SO); 00000200																				
5	NTP Administration	NTP Manager; 00200000																				
6	System Administration	System Manager; 02000000																				
7	User Management	User Manager; 20000000																				
All other user groups (3 and 4) can be used for customer-specific applications																						
Mechanism	<ul style="list-style-type: none">▪ RSA signature authentication mechanism▪ ECDSA signature authentication mechanism▪ HMAC password authentication mechanism																					

Property	Description
Attributes	<p>A string containing an assignment: <code><AttrName>=<AttrValue></code> Examples:</p> <ul style="list-style-type: none"> <code>CXI_GROUP=msk*</code> or <code>CXI_GROUP=*_mbk</code> This attribute defines that the cryptographic users they have been assigned to are permitted to generate, use, delete, export and import keys belonging to the CXI groups msk_1, msk_2, msk3 and so on, or to the groups A_mbk, B_mbk and so on. A concatenation of two attributes, e.g., <code>CXI_GROUP=msk*</code>, <code>CXI_GROUP=*_mbk</code> is not supported. <code>CXI_GROUP=SLOT_XXXX</code> This attribute must be set for PKCS#11 users of PKCS#11 slot X <p>The meaning of the Attributes depends on the application loaded into the device. The permission of a user may be restricted to a group of objects (keys, configuration and storage objects) that matches the user attribute <code>CXI_GROUP</code>. This means, a user is only allowed to access keys within his key group, and has no access to keys from other groups. If the user group attribute contains wildcards ('?' or '*'), the user may access multiple key groups; as example 2 above shows, <code>CXI_GROUP=msk*</code> allows the user to access msk01 as well as msk02. If a user was created without a user group attribute, he is only allowed to access global objects that do not belong to any group of objects. If a user was created with the attribute <code>CXI_GROUP=*</code>, he is allowed to access all objects regardless of their group property. The defined value is assigned to the user as the attribute A[], e.g. <code>A[CXI_GROUP=SLOT_0000]</code>, and can be displayed by executing the <code>csadm ListUser</code> command.</p>
HashAlgo	<p>Hash algorithm to be used for RSA signature authentication, ECDSA signature authentication or HMAC password authentication mechanism in case that not the default hash algorithm shall be used for this user, which is SHA-256 for RSA signature authentication, ECDSA signature authentication and HMAC password authentication. The value can be one of the following algorithms:</p> <ul style="list-style-type: none"> SHA-1, SHA-224, SHA-256, SHA-384 or SHA-512 MD5 RIPEMD-160 <p>The defined value is assigned to the user as the attribute <code>H[]</code>, for example, <code>H[SHA-256]</code>. In FIPS mode, only the default hash algorithms are supported.</p>

The creation of new users also requires an authentication:

- In general, authentication by one or two users with the permission to manage users (i.e., authentication level 2 in user group 7) is required.
- If a user with administrative rights shall be created: additionally, the authentication of one user with the permission to administrate the device is required (i.e., authentication level 2 in user group 7 and level 1 in user group 6).

For this reason, one initial user ADMIN is preconfigured and uses the RSA-Signature authentication mechanism with the Initialization Key. ADMIN has the exclusive permission of user management and administration (22000000) and does not require a second user to authenticate administrative commands.

The user ADMIN may be deleted from the user database only if one or more other users have been created, who – alone or in combination – possess the minimum permission to create users, including a user with administrative rights.



The user management of the device checks if the sum of the permissions of the remaining users with signature authentication mechanism (RSA, ECDSA) still allows user management and the creation of a user with administrative rights (i.e., resulting permission ≥ 21000000) and denies the deletion of a user eventually.

By this means users can never- accidentally - lock out themselves from the device.



Certain types of shell processes treat certain characters (commas, colons, semi-colons) differently. If the execution of a csadm command fails with an error message from the shell about an illegal parameter format, quoting parameter values may be necessary.

4.5.1 AddUser

With this command, a new user is added to the user database using RSA Signature, ECDSA Signature or HMAC Password as authentication mechanism. During the execution of this command, the public part of the RSA key is read from the smartcard or key file and stored in the cHSM's user database. See section [Password Authentication](#) for further details on the available mechanisms.

With this command, a new user is added to the user database using RSA Signature, ECDSA Signature or HMAC Password as authentication mechanism. It is impossible to create a user

with a user attribute 'H', and therefore it is impossible to change the default hashing algorithm for the authentication mechanism. The RSA key size has to be a minimum of 1024 bit.

Syntax	<code>csadm [Dev=<device>] <Authentication> AddUser=<user>,<permission>,<mechanism>,<auth_token></code>
---------------	--

Authentication	Permission level 2 in user group 7 (20000000)
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<user>	User name
<permission>	The user's permissions in 8 digits, each representing a permission level in a specific user group A list of attributes, put into curly braces {}, can be appended to the permission digits, see table with user properties in section 2021-0037 Commands for User Management and example below.
<mechanism>	<p>The authentication mechanism of the user</p> <ul style="list-style-type: none"> <code>hmacpwd</code> Authentication mechanism 'HMAC Password' <p>If a user with HMAC password authentication is created, this user cannot perform commands he/she needs an authentication for (except for the <code>csadm ChangeUser</code> command). The <code>csadm ListUser</code> command shows this user with the <code>I[1]</code> attribute value. To enable the user to perform commands he/she needs an authentication for, the user must change his/her credentials. To do so, if the user is a PKCS#11 Security Officer (SO) or a PKCS#11 normal user, perform the <code>p11tool2 SetPIN</code> command. Otherwise, this user must perform the <code>csadm ChangeUser</code> command. It is important that the changes are performed by the user himself/herself and not, for example, by an administrator. The <code>csadm ListUser</code> command then shows this user with the <code>I[0]</code> attribute value.</p> <ul style="list-style-type: none"> <code>rsasign</code> Authentication mechanism 'RSA Signature' <code>ecdsa</code> Authentication mechanism 'ECDSA Signature'

Parameter	Description
<auth_token>	<p>The authentication token of the user, which is a key specifier or a password, depending on the user's authentication mechanism:</p> <ul style="list-style-type: none"> ▪ Mechanism is <code>rsasign</code> / <code>ecdsa</code> Key specifier where the public part of the user's RSA or ECDSA key should be loaded from a smartcard specifier or a key file. In case of <code>rsasign</code> or <code>ecdsa</code>, the name of a non-default hash algorithm but into curly braces <code>{ }</code> can be appended or prepended to the key specifier. <p>ECDSA keys must be based on one of the FIPS 140-3-allowed curves, see <i>Built-in Elliptic Curves</i> in the <i>u.trust Anchor FIPS 140-3 - Containerized Hardware Security Module (cHSM) - Administration Manual</i>.</p> <ul style="list-style-type: none"> ▪ Mechanism is <code>hmacpwd</code> The user's password with a length between 8 and 1024 characters. <p>If the password is given as <code>ask</code>, hidden password entry will be performed which is strongly recommended. The name of a non-default hash algorithm put into curly braces <code>{ }</code> can be appended or prepended to the password.</p>

Example	<p>Adding a user with the authentication mechanism 'RSA Signature': <code>csadm LogonSign=ADMIN,:cs2:cjo:USB0 AddUser=Sven,01000000,rsasign,{SHA-256}key1.key</code></p> <p>Adding a user with the authentication mechanism 'HMAC Password' and hidden password entry: <code>csadm LogonSign=ADMIN,:cs2:cjo:USB0 AddUser=Sven,020{CXI_GROUP=SLOT_0004},hmacpwd,{SHA-512}ask</code></p>
----------------	--

Output	Upon successful execution of the command, no output is given.
---------------	---

4.5.1.1 Adding PKCS#11 Security Officers with RSA Signature Authentication with a Smartcard

Prerequisites

- The name of a Security Officer must be `SO_xxxx` with `xxxx` being a 2-byte decimal representation of the key group/PKCS#11 slot ID. The name may range from `SO_0000` to `SO_9999`.
- The Security Officer must have at least permission 00000200.
- The Security Officer must be assigned to the appropriate key group/PKCS#11 slot. The slot specifier may range from `SLOT_0000` to `SLOT_9999`.
- Only HMAC password authentication is supported.

However, if you want to enforce having two users with RSA signature authentication with a smartcard to perform actions as a Security Officer, there is a solution. This section describes how to create the needed users. In total, the following three users are needed:

- A Security Officer named `SO_0000` (0000 for PKCS#11 slot 0) with permission 00000000 and HMAC password authentication.
The purpose of this user is to provide a user with the needed name `SO_0000` and the HMAC password authentication enforced by PKCS#11. The permission must be 00000000 so that this user cannot perform any action at all and other users providing the needed permissions and the desired RSA signature authentications with a smartcard are needed.
- A Security Officer named, for example, `SOSC1_0000` with permission 00000100 and RSA signature authentication with a smartcard
This user provides the desired RSA signature authentication with a smartcard and the first half of the needed permission.
- A Security Officer named, for example, `SOSC2_0000` with permission 00000100 and RSA signature authentication with a smartcard
This user provides the desired RSA signature authentication with a smartcard and the second half of the needed permission.

If you log in all three users, the summarized permission of the three users is $00000000 + 00000100 + 00000100 = 00000200$.

Perform the following steps to create the needed users.

1. Create the Security Officer SO_0000.

Example:

```
csadm LogonSign=ADMIN,:cs2:cjo:USB0  
AddUser=SO_0000,000000000{CXI_GROUP=SLOT_0000},hmacpwd,123456
```

2. Create the Security Officer SOS1_0000.

Example:

```
csadm LogonSign=ADMIN,:cs2:cjo:USB0  
AddUser=SOS1_0000,00000100{CXI_GROUP=SLOT_0000},rsasign,:cs2:cjo:USB0
```

3. Create the Security Officer SOS2_0000.

Example:

```
csadm LogonSign=ADMIN,:cs2:cjo:USB0  
AddUser=SOS2_0000,00000100{CXI_GROUP=SLOT_0000},rsasign,:cs2:cjo:USB0
```

4. Now you can perform a command as a Security Officer logging in the three users.

Example:

```
csadm LogonPass=SO_0000,123456 LogonSign=SOS1_0000,:cs2:cjo:USB0  
LogonSign=SOS2_0000,:cs2:cjo:USB0 ...
```

4.5.2 BackupUser

This command backs up all users that are currently set up on the cHSM into a file. Each user entry in the created backup file is encrypted with the cHSM's Master Backup Key (MBK) and therefore can be handled without additional security measures.

Perform the `csadm MBKListKeys` command to determine which Master Backup Key (MBK) is currently in use in MBK slot 3 by the device. This MBK is used by the `csadm BackupUser` command to protect the backup file to be generated. If the MBK in MBK slot 3 is the autogenerated MBK named `AUTO-GEN`, the `csadm BackupUser` command cannot be performed. Import a different MBK into MBK slot 3 using the `csadm MBKImportKey` command.



The user 'ADMIN' will not be backed up.



It is important to note down which MBK has been used because for a successful restoring of this backup file at a later date it is necessary that the same MBK is in MBK slot 3. Otherwise, for example, after the execution of a `csadm MBKImportKey` command or after an MBK rollover, the backup file is inaccessible.

If user data is backed up using an old firmware not supporting the `I[]` attribute and this user data is restored using a new firmware supporting the `I[]` attribute, the restored users do not have the `I[]` attribute, i.e., the restored users do not need to change their credentials.

If users having the `I[]` attribute are backed up and restored, the restored users have the `I[]` attribute in the state they had before the backup.

You cannot downgrade the `I[]` attribute, i.e., perform a `csadm BackupUser` command for a user having the `I[]` attribute and then perform a `csadm RestoreUser` command on an old firmware not supporting the `I[]` attribute. This procedure would cause a `Bad user attribute` error (B0830017).

For more details about the `I[]` attribute, see 2009-0003 ListUser.

As of SecurityServer 6.0.0, a new user backup structure/format has been introduced.

- For creating and restoring backups, it is recommended to use a csadm version that matches the firmware version.
- Backups with the format earlier than 6.0.0 are still possible to be restored, except in FIPS mode because FIPS does not allow the old format.
- Restoring a backup with the new format into firmware version < 6.0.0 is not possible. Generally, restoring backups of a newer firmware version into older firmware is not supported.
- As of version 6.0.0, customers must use the `csadm BackupUser` command to back up users, because `csadm ... BackupDatabase=user.db` is not supported anymore.

Syntax

`csadm [Dev=<device>] <Authentication> BackupUser=<file>[,<flags>]`

Authentication

Permission level 2 in user group 7 (20000000)

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<file>	Path and file name of the backup file to be created
<flags>	Optional flags Overwrite: Backup file will be overwritten if already existing If the overwrite flag is not given and the backup file already exists the backup will not be performed and the existing backup file will not be overwritten.

Example	csadm LogonSign=ADMIN,:cs2:cjo:USB0 BackupUser=d:\temp\cs123456-20051212.ubk,overwrite
----------------	--

Output	Upon successful execution of the command, no output is given.
---------------	---

4.5.3 ChangeUser



A user is not allowed to change their authentication mechanism, permission or flags.



To change a user with authentication mechanism RSA Signature on a cHSM, the RSA key size has to be minimum 2048 bit.

Syntax	csadm [Dev=<device>] <Authentication> ChangeUser=<user>,<auth_token>
---------------	---

Authentication	The command must be authenticated by the existing user according to their authentication mechanism.
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Parameter	Description
<user>	Existing user name
<auth_token>	<p>The user's new authentication token (key specifier or password) depending on the user's authentication mechanism:</p> <p>'RSA Signature' / 'ECDSA Signature': Key specifier where the public part of the user's RSA or ECDSA key should be loaded from a smartcard specifier (e.g. <code>:cs2:cjo:USB0</code>) or a key file (e.g. <code>my.key</code>). ECDSA keys must be based on one of the FIPS 140-3-allowed curves, see <i>Built-in Elliptic Curves in the u.trust Anchor FIPS 140-3 - Containerized Hardware Security Module (cHSM) - Administration Manual</i>.</p> <p>'HMAC Password': The user's new password (length between 8 and 1024 characters). If the password is given as <code>ask</code>, hidden password entry will be performed, which is strongly recommended.</p>

Example	<ul style="list-style-type: none"> ▪ <code>csadm LogonSign=sven,:cs2:cjo:USB0 ChangeUser=sven,:cs2:cjo:USB0</code> ▪ <code>csadm LogonPass=sven,swordfish ChangeUser=sven,sesame</code> ▪ <code>csadm LogonPass=nils,ask ChangeUser=nils,ask</code>
----------------	--

Output	Upon successful execution of the command, no output is given.
---------------	---



When the credentials of a user are changed and the new credentials are equal to the old credentials, then the system returns an error and displays the message: "The specified user credentials are already in use by the specified user."

Changing another user's HMAC password

If a user with user management rights changes the authentication credentials (using the `csadm ChangeUser` command) of another user who has HMAC password authentication, this other user cannot perform commands he/she needs an authentication for (except for the `csadm ChangeUser` command). The `csadm ListUser` command shows this user with the `I[1]` attribute value. To enable the user to perform commands he/she needs an authentication for, the user must change his/her credentials. To do so, this user must perform the `csadm ChangeUser` command. If the user is a PKCS#11 Security Officer (SO) or a PKCS#11 normal user, he/she may perform the `p11tool2 SetPIN` command as an alternative. It is important that the changes are performed by the user himself/herself and

not, for example, by an administrator. The `csadm ListUser` command then shows this user with the `I[0]` attribute value.

Example procedure:

1. Show which users are available.

```
csadm Dev=4001@194.167.1.3 ListUser
```

Example output:

Name	Permission	Mechanism	Attributes
ADMIN	22000000	RSA sign	Z[0]I[0]
test	22000000	HMAC passwd	Z[0]I[0]

2. Verify that the test user is capable of performing a command he/she needs an authentication for, for example, by performing a `csadm MBKListKeys` command.

```
csadm Dev=4001@194.167.1.3 LogonPass=test,12345678 MBKListKeys
```

Example output:

slot	name	len	algo	type	k	generation	date	key	check	value
3	<NoName>	32	AES	XOR	2	2007/08/06	10:35:50	106B5E4E84031BDE		
7	AUTO-GEN	32	AES	SHARE	1	2023/08/07	10:11:11	10785E4E84032ADE		

3. Let the admin user change the test user's credentials.

```
csadm Dev=4001@194.167.1.3 LogonSign=ADMIN,:cs2:cjo:USB0
ChangeUser=test,11223344
```

4. The test user now is shown with the `I[1]` attribute, i.e., he/she cannot perform commands he/she needs an authentication for (except for the `csadm ChangeUser` command).

```
csadm Dev=4001@194.167.1.3 ListUser
```

Example output:

Name	Permission	Mechanism	Attributes
ADMIN	22000000	RSA sign	Z[0]I[0]
test	22000000	HMAC passwd	Z[0]I[1]

5. This is why the test user cannot perform a command he/she needs an authentication for, for example, a `csadm MBKListKeys` command.

```
csadm Dev=4001@194.167.1.3 LogonPass=test,11223344 MBKListKeys
```

Example output:

```
Error B0830091
CryptoServer module CMDS, Command scheduler
The user credentials need to be updated
```

6. The test user must change his/her credentials.

```
csadm Dev=4001@194.167.1.3 LogonPass=test,11223344
ChangeUser=test,12345678
```

7. The test user is now shown with the `I[0]` attribute, i.e., he/she can perform commands he/she needs an authentication for.

```
csadm Dev=4001@194.167.1.3 ListUser
```

Example output:

Name	Permission	Mechanism	Attributes
ADMIN	22000000	RSA sign	Z[0]I[0]
test	22000000	HMAC passwd	Z[0]I[0]

8. Verify that the test user can perform commands he/she needs an authentication for, for example, by performing a `csadm MBKListKeys` command.

```
csadm Dev=4001@194.167.1.3 LogonPass=test,12345678 MBKListKeys
```

Example output:

slot	name	len	algo	type	k	generation	date	key	check	value
3	<NoName>	32	AES	XOR	2	2007/08/06	10:35:50	106B5E4E84031BDE		
7	AUTO-GEN	32	AES	SHARE	1	2023/08/07	10:11:11	10785E4E84032ADE		

4.5.4 DeleteUser

This function deletes a user from the user database.

In order to prevent the device from getting non-administrable, the sum of the permissions of the remaining users, who use a signature-based authentication mechanism, has to be at least level 2 in user group 7 and level 1 in group 6. If this is not the case, the function will refuse execution and return an error code.

Syntax	<code>csadm [Dev=<device>] <Authentication> DeleteUser=<user></code>
---------------	--

Authentication	Permission level 2 in user group 7 (20000000)
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Parameter	Description
<user>	Existing user name

Example	csadm LogonSign=ADMIN,:cs2:cjo:USB0 DeleteUser=sven
----------------	---

Output	Upon successful execution of the command, no output is given.
---------------	---

4.5.5 GetMaxAuthFails

This command displays the unallowed number `n` of consecutive failed authentication attempts for each user.

Syntax	csadm [Dev=<device>] GetMaxAuthFails
---------------	--------------------------------------

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	csadm Dev=4001@192.168.1.1 GetMaxAuthFails
----------------	--

Output	Upon successful execution, the command returns <code>n</code> (0...255), the unallowed number <code>n</code> of consecutive failed authentication attempts for each user. If this number is reached the user is blocked and unable to authenticate towards the cHSM anymore.
---------------	--

4.5.6 ListUser

This command lists all existing users from the `user.db` database.

Syntax	csadm [Dev=<device>] ListUser
---------------	-------------------------------

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<user>	Existing user name

Example	csadm Dev=4001@192.168.1.1 ListUser
----------------	-------------------------------------

Output	Upon successful execution of the command, a list of all existing users is given. The initial user ADMIN is always displayed even if the database <code>user.db</code> is not yet present.			
	Name	Permission	Mechanism	Attributes
	ADMIN	22000000	RSA Sign	Z[[0]H[SHA-256]
	sm	00000022	HMAC passwd	Z[1]I[0]
	Sven	22000022	HMAC passwd	I[0]
	SO_0007	00000201	HMAC passwd	
	A[CXI_GROUP=SLOT_0007]L[0007]I[0]			
	CryptU	00000002	RSA sign	A[CXI_GROUP=msk]

Parameter	Description
A[]	Application-depending attributes, for example, A[CXI_GROUP=SLOT_0007]
H[]	Non-default hash algorithms used for the authentication mechanism of that user, for example, H[SHA-256]. The H[] attribute is only shown if this attribute has been explicitly set when the user has been added to the device.
Z[]	Counter for the number of consecutively failed authentication attempts marked by the tag Z, e.g. Z[1]. The attribute Z is set for every user on first authentication attempt to the default Z[0]. If the authentication succeeded, Z remains Z[0], otherwise the counter is incremented by one.
L[]	PKCS#11 slot label which is assigned to the slot's Security Officer (SO) during slot initialization. By default, this attribute is set to <code>CryptoServer PKCS11 Token</code> . It can be changed by the default device Administrator ADMIN or a user administrator during PKCS#11 slot initialization using one of the PKCS#11 administration tools – P11CAT or p11tool2. The PKCS#11 slot must not be confused with the MBK slot.

Parameter	Description
I[]	<p>I[] only applies to users with HMAC password authentication.</p> <ul style="list-style-type: none"> I[1] The user cannot perform commands he/she needs an authentication for (except for the <code>csadm ChangeUser</code> command), i.e., one of the following cases applies: <ul style="list-style-type: none"> Initial state when the user has been created. Another user (e.g., an administrator) has changed the user's password. <p>To enable the user to perform commands he/she needs an authentication for, the user must change his/her credentials. To do so, if the user is a PKCS#11 Security Officer (SO) or a PKCS#11 normal user, perform the <code>p11tool2 SetPIN</code> command. Otherwise, this user must perform the <code>csadm ChangeUser</code> command. It is important that the changes are performed by the user himself/herself and not, for example, by an administrator.</p> <ul style="list-style-type: none"> I[0] The user has changed his/her own password. The user can perform commands he/she needs an authentication for.

Table 12: Meaning of the return parameters of `csadm ListUser`

If user data is backed up using an old firmware not supporting the I[] attribute and this user data is restored using a new firmware supporting the I[] attribute, the restored users do not have the I[] attribute, i.e., the restored users do not need to change their credentials.

If users having the I[] attribute are backed up and restored, the restored users have the I[] attribute in the state they had before the backup.

You cannot downgrade the I[] attribute, i.e., perform a `csadm BackupUser` command for a user having the I[] attribute and then perform a `csadm RestoreUser` command on an old firmware not supporting the I[] attribute. This procedure would cause a `Bad user attribute` error (B0830017).



The attribute L is only set when the user with role SO is created during slot initialization with one of the PKCS#11 administration tools – P11CAT or p11tool2. It cannot be set if you create the SO for a specific PKCS#11 slot with CAT or csadm.

4.5.7 RestoreUser

This command restores the users in the `user.db` database on the device from a backup file created by the `csadm BackupUser` command.



The login status of the user here only refers to the current session of the authentication of the `csadm BackupUser` command. If a user is logged in by using another tool, for example, via p11CAT, this user can be overwritten.



The same Master Backup Key (MBK) that has been used to create the backup file must be stored on the device.



To find out whether a user already exists in the `user.db` database on the device, perform the `csadm ListUser` command. Do not try to delete the user.

If user data is backed up using an old firmware not supporting the `I[]` attribute and this user data is restored using a new firmware supporting the `I[]` attribute, the restored users do not have the `I[]` attribute, i.e., the restored users do not need to change their credentials.

If users having the `I[]` attribute are backed up and restored, the restored users have the `I[]` attribute in the state they had before the backup.

You cannot downgrade the `I[]` attribute, i.e., perform a `csadm BackupUser` command for a user having the `I[]` attribute and then perform a `csadm RestoreUser` command on an old firmware not supporting the `I[]` attribute. This procedure would cause a `Bad user attribute` error (B0830017).

For more details about the `I[]` attribute, see *ListUser*.

As of SecurityServer 6.0.0, a new user backup structure/format has been introduced.

- For creating and restoring backups, it is recommended to use a csadm version that matches the firmware version.
- Backups with the format earlier than 6.0.0 are still possible to be restored, except in FIPS mode because FIPS does not allow the old format.
- Restoring a backup with the new format into firmware version < 6.0.0 is not possible. This procedure would cause an `Illegal length of command block` error (B0830008). Generally, restoring backups of a newer firmware version into older firmware is not supported.

- As of version 6.0.0, customers must use the `csadm BackupUser` command to back up users, because `csadm ... BackupDatabase=user.db` is not supported anymore.



As of u.trust Anchor FIPS 140-3 6.0.0, a new firmware and a new user data backup format are applied. As a consequence, if you have backed up user data using an u.trust Anchor < 6.0.0 (i.e. using the old firmware with the old user data backup format), this data backup cannot be restored in one step by u.trust Anchor FIPS 140-3 >= 6.0.0.

Instead, proceed as follows:

1. Restore the user data backup using u.trust Anchor non-FIPS >= 6.0.0.
2. Create a new user data backup using u.trust Anchor non-FIPS >= 6.0.0 (i.e. using the new firmware with the new user data backup format).
3. Restore the new user data backup using u.trust Anchor FIPS 140-3 >= 6.0.0 (i.e. using the new firmware with the new user data backup format).

If users cannot be restored, for example because they use a HASH that is no longer supported for HMAC authentication, they will be skipped during the restore.

Administrators get an output, how many users have been restored and listing those who could not be restored.

For example, skipping/not restoring invalid users prevents users who can no longer log in from ending up in the database.

Example output RestoreUser

```
Restored 5 out of 8 users in backup.
Skipped user user 3 (B0890007)
Skipped user user 7 (B0890007)
Skipped user Admin (User is currently logged in)
```

Syntax

```
csadm [Dev=<device>] <Authentication> RestoreUser=<file>[,<flags>]
```


Authentication

Permission level 2 in user group 7 (20000000)

<i>Parameter</i>	<i>Description</i>
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<file>	Path and file name of the backup file

Parameter	Description																
<flags>	The audit log entries <code>Delete User</code> and <code>Restore User</code> mentioned below are actually as follows: <code>FC:0x083 SFC:0x05 Delete User ‘<user name>’ [error code]</code> and <code>FC:0x083 SFC:0x0D Restore User ‘<user name>’ (<short user name>, <authentication mechanism ID> <user’s permission> <access properties>) [error code]</code> <code><access properties></code> contains a comma-separated list of strings of the form <code><name>=<value></code> . Examples: <code>CXI GROUP=SLOT_0001</code> <code>CXI_GROUP=test</code> See section <code>OS_AUDIT_CLASS_USER</code> in the <i>u.trust Anchor - Containerized Hardware Security Module (cHSM) - Administration Manual</i> for details.																
	add (default)																
	<table><tr><th>Scenario</th><th>Action</th><th>Audit Log Entry</th><th>Output</th></tr><tr><td>User in the backup file does not exist yet in the <code>user.db</code> database</td><td>User is added</td><td><code>Restore User</code></td><td>User <code><username></code> added to <code>user.db</code></td></tr><tr><td>User in the backup file already exists with in the <code>user.db</code> database</td><td>None</td><td>None</td><td>User <code><username></code> not added to <code>user.db</code> because the account already exists</td></tr><tr><td>User exists in the <code>user.db</code> database but not in the in the backup file</td><td>None</td><td>None</td><td>None</td></tr></table>	Scenario	Action	Audit Log Entry	Output	User in the backup file does not exist yet in the <code>user.db</code> database	User is added	<code>Restore User</code>	User <code><username></code> added to <code>user.db</code>	User in the backup file already exists with in the <code>user.db</code> database	None	None	User <code><username></code> not added to <code>user.db</code> because the account already exists	User exists in the <code>user.db</code> database but not in the in the backup file	None	None	None
	Scenario	Action	Audit Log Entry	Output													
	User in the backup file does not exist yet in the <code>user.db</code> database	User is added	<code>Restore User</code>	User <code><username></code> added to <code>user.db</code>													
	User in the backup file already exists with in the <code>user.db</code> database	None	None	User <code><username></code> not added to <code>user.db</code> because the account already exists													
	User exists in the <code>user.db</code> database but not in the in the backup file	None	None	None													
	overwrite																
	<table><tr><th>Scenario</th><th>Action</th><th>Audit Log Entry</th><th>Output</th></tr><tr><td>User in the backup file does not exist yet in the <code>user.db</code> database</td><td>User is added</td><td><code>Restore User</code></td><td>User <code><username></code> added to <code>user.db</code></td></tr><tr><td>User in the backup file already exists with in the <code>user.db</code> database and user is not logged in</td><td>User is overwritten</td><td><code>Delete User</code> <code>Restore User</code></td><td>User <code><user name></code> replaced in <code>user.db</code></td></tr><tr><td>User in the backup file already exists with in the <code>user.db</code> database and user is logged in</td><td>None</td><td>None</td><td>User <code><user name></code> not replaced in <code>user.db</code> because the user is currently logged in</td></tr></table>	Scenario	Action	Audit Log Entry	Output	User in the backup file does not exist yet in the <code>user.db</code> database	User is added	<code>Restore User</code>	User <code><username></code> added to <code>user.db</code>	User in the backup file already exists with in the <code>user.db</code> database and user is not logged in	User is overwritten	<code>Delete User</code> <code>Restore User</code>	User <code><user name></code> replaced in <code>user.db</code>	User in the backup file already exists with in the <code>user.db</code> database and user is logged in	None	None	User <code><user name></code> not replaced in <code>user.db</code> because the user is currently logged in
	Scenario	Action	Audit Log Entry	Output													
User in the backup file does not exist yet in the <code>user.db</code> database	User is added	<code>Restore User</code>	User <code><username></code> added to <code>user.db</code>														
User in the backup file already exists with in the <code>user.db</code> database and user is not logged in	User is overwritten	<code>Delete User</code> <code>Restore User</code>	User <code><user name></code> replaced in <code>user.db</code>														
User in the backup file already exists with in the <code>user.db</code> database and user is logged in	None	None	User <code><user name></code> not replaced in <code>user.db</code> because the user is currently logged in														

Parameter	Description			
	Scenario	Action	Audit Log Entry	Output
	User exists in the <code>user.db</code> database but not in the in the backup file	None	None	None
	replace			
	User in the backup file does not exist yet in the <code>user.db</code> database	User is added	Restore User	User <user name> added to user.db
	User in the backup file already exists with in the <code>user.db</code> database	User is overwritten	Delete User Restore User	User <user name> replaced in user.db
	Single user manager is logged in and this user manager is included in the backup file or several user managers are logged in and at least one of these user managers is included in the backup file	error number != 0	None	RestoreUser in 'replace' mode is not supported when logged-in user manager(s) shall be restored.
	User exists in the <code>user.db</code> database but not in the in the backup file	User is deleted	Delete User	User <user name> removed from HSM
	 If the deleted user a single logged-in user manager, deleting this user manager is the last operation of the procedure, because deleting him terminates the Secure Messaging session. If you have a new device, the ADMIN user is the only existing user. If the backup file does not include the ADMIN user and you perform the <code>csadm RestoreUser</code> command, all users from the backup file are added to the <code>user.db</code> database and then the ADMIN user is deleted.			

Parameter	Description
	 If a single user manager is logged in and this user manager is included in the backup file, the <code>csadm RestoreUser</code> command returns an error (<code>error number != 0</code>). The restore operation is aborted and not a single user account is restored. The same applies if several user managers are logged in and at least one of these user managers is included in the backup file.

Example	<code>csadm LogonSign=ADMIN,:cs2:cjo:USB0 RestoreUser=d:\temp\cs123456-20051212.ubk,overwrite</code>
----------------	--

Output	Upon successful execution of the command, an indication of the executed action is returned as stated in the flag table above.
---------------	---

4.5.8 SetMaxAuthFails

This command defines the maximum number (n) of consecutive failed authentication attempts for each user. Optionally, the maximum can be set for users with HMAC password authentication only.

If this maximum number is reached the user is blocked. A blocked user is not able to authenticate towards the device anymore.

A blocked user can be unblocked by a user with user management rights (minimum permission 2 in the user group 7; 20000000) who is logged on to the device.

`SetMaxAuthFails` is a global configuration, i.e., it applies to all users except if the `pwdOnly` parameter is set.

Syntax	<code>csadm [Dev=<device>] <Authentication> SetMaxAuthFails=<n>[,pwdOnly]</code>
---------------	--

Authentication	Permission level 2 in user group 7 (20000000)
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<n>	The maximum number of consecutive failed authentication attempts for each user (0 - 255.) A value of 0 means that an unlimited number of consecutive failed authentication attempts is allowed for each user. The default for n is 0.
<pwdOnly>	<code>pwdOnly</code> is available only for csadm version 2.5.1 or higher. If <code>pwdOnly</code> is set, the maximum number of consecutive failed authentication attempts is only applied to all users using HMAC password authentication. If <code>pwdOnly</code> is set and a user uses RSA or ECDSA signature authentication or RSA smartcard authentication, an unlimited number of consecutive failed authentication attempts is allowed for this user. The setting <code>pwdOnly</code> is equivalent to <code>pwdOnly=1</code> , <code>pwdOnly=y</code> , <code>pwdOnly=yes</code> and <code>pwdOnly=0</code> .
Example	<pre>csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,:cs2:cjo:USB0 SetMaxAuthFails=3 csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,:cs2:cjo:USB0 SetMaxAuthFails=3,pwdOnly csadm LogonPass=Sven,ask SetMaxAuthFails=4 csadm LogonPass=Sven,ask SetMaxAuthFails=4,pwdOnly</pre>
Output	Upon successful execution of the command, no output is given.

4.6 Commands for Managing the User Authentication Keys

In this chapter the csadm commands for generation, administration and backup of user authentication keys (or tests keys) are described. The generated keys may be stored either on a smartcard or in a keyfile.

The following command group contains internal functions of csadm. No connection to a device will be established.



Certain types of shell processes treat certain characters (commas, colons, semi-colons) differently. If the execution of a csadm command fails with an error message from the shell about an illegal parameter format, quoting parameter values may be necessary.

4.6.1 BackupKey

This command reads a key from a keyfile, splits it into two XOR parts and saves the parts on two smartcards for backup matters.

This command is executed locally without a connection to a device. Therefore, the state or mode of any underlying device is irrelevant for the command execution, and no authentication at any device is necessary.

If smartcards are used, a PIN pad can be connected to the computer where the csadm tool is running (USB port) or to another computer.

Watch the display of the PIN pad for instructions on further command processing.



A backup generated by the `csadm BackupKey` command is stored on smartcards. An MBK rollover is irrelevant for this backup. This backup can be restored after an MBK rollover.

Syntax	<code>csadm [KeyType=<keytype>] PrvKey=<keyfile> BackupKey=<cardspec></code>
Parameter	Description
<keytype>	Key type: RSA (default) or EC
<keyfile>	File where the key is stored (*.key)
<cardspec>	Smartcard specifier of the back-up smartcards
Example	<code>csadm PrvKey=C:\my_keys\prvkey1.key BackupKey=:cs2:cjo:USB0 csadm KeyType=EC PrvKey=C:\my_keys\myprvkey.key#mypassword BackupKey=:cs2:cjo:USB0</code>
Output	Upon successful execution of the command, no output is given.

4.6.2 CopyBackupCard

This command copies one XOR-half of a key-backup (backup of user's authentication key) from one smartcard to another.

This command is executed locally without a connection to a cHSM. Therefore the state or mode of any underlying cHSM is irrelevant for the command execution, and no authentication is necessary.



A PIN pad including a smartcard reader has to be connected to the computer where the csadm tool is running. Watch the PIN pad's display for instructions on further command processing.

Syntax	csadm [KeyType=<keytype>] CopyBackupCard=<keyspec>
Parameter	Description
<keytype>	Key type. RSA (default) or EC. The csadm CopyBackupCard command copies an RSA share or an EC share, which is specified by the <keytype> parameter. The corresponding action initiated by Java-based GUI CryptoServer Administration Tool (CAT) copies an RSA share and an EC share at the same time.
<keyspec>	Key specifier of the backup-smartcards, source, and target
Example	csadm KeyType=EC CopyBackupCard=:cs2:cjo:USB0
Output	Upon successful execution of the command, no output is given.

4.6.3 ChangePassword

This command changes the password of an encrypted keyfile (*.key).

The command can also be used to convert a plaintext keyfile into an encrypted keyfile and vice versa:

- The command can be used to encrypt a keyfile that has been a plaintext keyfile before. In this case, the parameter Password= has to be omitted and only the NewPassword= parameter has to be given.
- The command can also be used to decrypt a keyfile, i. e. to turn an encrypted keyfile into a plaintext keyfile. In this case, the parameter NewPassword= has to be omitted and only the Password= parameter has to be given.



This command is executed locally without a connection to a cHSM. Therefore, the state or mode of any underlying cHSM is irrelevant for the command execution, and no authentication is necessary.



The usage of hidden password entry is strongly recommended. If hidden password entry is used, the csadm will ask for the keyfile's old and new password separately (i.e. a request 'Enter Passphrase:' respectively 'Enter New Passphrase:' follows in the next command line) and hide the entrance on the monitor by the display of default characters.

Syntax

```
csadm [KeyType=<keytype>] NewPassword=<new_password>  
ChangePassword=<keyfile[#old_password]>
```

Parameter	Description
<keytype>	RSA (Default) EC
<old_password>	The old password of the encrypted key file If no password is given (or password is 'ask'), hidden password entry is possible, which is strongly recommended.
<new_password>	The new password for the encrypted key file If no password is given (or password is 'ask'), hidden password entry is possible, which is strongly recommended.
<keyfile>	The encrypted key file (*.key) .

Example

```
csadm Password=swordfish NewPassword=shark  
ChangePassword=c\keys\rsa_key_enc.key
```

Output

Upon successful execution of the command, no output is given.



When the credentials of a user are changed and the new credentials are equal to the old credentials, then the system returns an error and displays the message: "The specified user credentials are already in use by the specified user."

4.6.4 ChangePIN

This command changes the PIN of a given smartcard. A PIN pad including a smartcard reader must be used for this command. The PIN pad has to be connected to a USB port of that computer where the csadm tool is running.

This command is executed locally without a connection to a cHSM. Therefore, the state or mode of any underlying cHSM is irrelevant for the command execution and no authentication at any cHSM is necessary.



Watch the PIN pad's display: Enter the old PIN first, then enter a new PIN and confirm the new PIN.

Syntax	
	csadm ChangePIN=<keyspec>

Parameter	Description
<keyspec>	Specifier for smartcard type, PIN pad type and USB port

Example	
	csadm ChangePIN=:cs2:cjo:USB0

Output	
	Upon successful execution of the command, no output is given.

4.6.5 GetCardInfo

This command reads information about keys eventually stored on a smartcard: The info records of the user's authentication key (e.g. Default Administrator Key) and back-up key will be scanned and output.

A smartcard can contain an RSA key, an elliptic-curve cryptography key, an RSA key backup share, an elliptic key backup share, and an MBK backup share at the same time.

- **RSA-Key**

An RSA key. Use the csadm `GenKey` command to generate it. The default value is `Utimaco IS GmbH / Init-Dev-1-Key` or `Utimaco IS GmbH / ADMIN-Key` (older

default value). Both default keys are the key for the default administrator (ADMIN user name).

- **ECC-Key**

An elliptic-curve cryptography key. Use the csadm `GenKey` command to generate it. The default value is `Utimaco IS GmbH / Default_EC_Key`.

- **MBK**

A backup share of a master backup key (MBK). Use the `csadm MBKGenerateKey` command to generate it. If you want to retrieve information only about MBK shares consider to perform the `csadm MBKCardInfo` instead of performing `csadm GetCardInfo`. The `csadm GetCardInfo` command is executed locally without a connection to a device. Therefore, the state or mode of any underlying device is irrelevant for the command execution, and no authentication at any device is necessary.

This command is executed locally without a connection to a cHSM. Therefore the state or mode of any underlying cHSM is irrelevant for the command execution, and no authentication is necessary.

Syntax	<code>csadm GetCardInfo=<keyspec></code>
Parameter	Description
<code><keyspec></code>	Key specifier of the smartcard
Example	<code>csadm GetCardInfo=:cs2:cjo:USB0</code>
Output	Upon successful execution, the command returns the info records of the user's authentication key (e.g. Default Administrator Key) and back-up key.

4.6.6 GenKey



This command is executed locally without a connection to the cHSM. Therefore, the state or mode of any underlying cHSM is irrelevant for the command execution, and no authentication to the cHSM is necessary.

With the `GenKey` command, you can generate an RSA or ECDSA key pair. The keys can either be generated and saved directly on a smartcard (recommended) or they can be stored in an encrypted or unencrypted keyfile.



Only use an unencrypted keyfile for testing purposes if security is not relevant.

- **Key generation and storage on smartcard (recommended):**

Your keys are generated on the smartcard and stored there. All operations with the private key are carried out on the card. The private key never leaves the card. Users need a PIN to access the smartcard.



The Java smartcard version must be 2.0.0.0 or later. If you want to store RSA keys longer than 2048 bits on the smartcard, Utimaco smartcard version 3.0.0.0 or higher is required. Use the `csadm GetCardInfo` command to check the version: If no output line `Utimaco Smartcard version` is shown, the smartcard does not support generating keys on the smartcard and storing them.

These keys cannot leave the smartcard and cannot be backed up.

Fallback users

Since these keys cannot be backed up, the owner must be issued with at least a second smartcard containing an authentication key that is assigned to a so-called "fallback user." This smartcard then can be used if the initial one is lost or broken.

For the additional smartcards, new users with the appropriate rights (fallback users) must be created and smartcards must be issued for them (for example `James`, `James_1`, ...).

This means that the person has then several smartcards which they can use for authentication. If the initial smartcard becomes unusable for any reason, the owner can log in as a fallback user using the associated smartcards.

The smartcards for the fallback users must be stored in a safe place!

- **Keys stored in keyfile - the keyfile stored on a smartcard:**

A keyfile `*.key` containing the keys is generated. You can then store the keyfile on a smartcard. For storing keys on smartcards see the `csadm SaveKey` command.

For backup purposes, you can store the keyfiles on several smartcards. The smartcards and the associated PINs must be stored in a safe place - if necessary, store these

separate from one another.

- **Keys stored in a keyfile:**


A keyfile `*.key` containing the keys is generated. The generated keyfile can be secured with a passphrase or created unencrypted (not recommended).

Where the keys are generated/stored is specified by a key specifier, see *Storage and Specification of RSA and ECDSA Keys for Authentication*.

The `GenKey` command is executed locally without a connection to the cHSM. Therefore, the state or mode of any underlying cHSM is irrelevant to the command execution, and no authentication to the cHSM is necessary.

Syntax	<code>csadm KeyType=[RSA EC] Key=<keyspec>[,<bitlength>],<owner></code>
---------------	---

Parameter	Description
<keytype>	Type of the key: <ul style="list-style-type: none">▪ <code>RSA</code> (default)▪ <code>EC</code> for ECDSA

Parameter	Description
<keyspec>	<p>The key specifier determines where the user authentication keys are stored/generated (smartcard or keyfile).</p> <ul style="list-style-type: none"> Smartcard specifier, for example, <code>:cs2:cjo:USB0</code> The user authentication keys are generated on the smartcard and are stored on it. They cannot leave this smartcard and cannot be backed up. <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;">  A newly generated RSA key overwrites an already existing RSA key on the smartcard. A newly generated EC key overwrites an already existing EC key on the smartcard. </div> <ul style="list-style-type: none"> Storage location and name of the keyfile (<code>*.key</code>) If you want to create an encrypted keyfile you have to add a passphrase: <code>... *.key[#<passphrase>]</code> where <code><passphrase></code> is the passphrase used for protecting the keyfile. If you do not specify a passphrase, a plaintext keyfile is generated (not recommended). The passphrase can be specified in 2 ways: <ul style="list-style-type: none"> in plain text For example: <code>MyKey.key#mypwd</code> by adding the <code>#ask</code> string that triggers a hidden passphrase entry (highly recommended). You will be asked for the passphrase and have to confirm it. The passphrase will not be displayed when you enter it. For example: <code>... \MyKey.key#ask</code>
<bitlength>	Bit length of the generated RSA key, $512 \leq \text{bitlength} \leq 8192$. Default value: <code>2048</code> bit.
<curve>	<p>Name of the elliptic curve if the <code><KeyType></code> parameter is set to EC. Default value: <code>brainpoolP320t1</code>.</p> <p>If the <code><keyspec></code> parameter is set to a smartcard specifier, only the <code>brainpoolP320t1</code> value is supported for the <code><curve></code> parameter.</p>
<owner>	Owner or name of the key

Example	<ul style="list-style-type: none"> Generate RSA keys on the smartcard csadm <code>csadm GenKey=:cs2:cjo:USB0,"CHSM User"</code> Generates an encrypted keyfile that contains RSA keys. You will be asked for the passphrase. <code>csadm GenKey=C: \my_keys\testkey1.key#ask,2048,TestUser</code>
----------------	---

Output

If the command executes successfully, there is no output.

4.6.7 SaveKey (Standard)

This command reads a key from one storage medium and stores it to another storage medium. The key may either be the public or private part of a key pair or one half of a backup key set.

The following table shows the possible combinations of source and target mediums:

Key type	Source medium	Target medium
Public	keyfile	keyfile
	smartcard	keyfile
Private + Public	keyfile	keyfile or smartcard
Backup	keyfile	keyfile or smartcard
	smartcard	keyfile or smartcard

It is not possible to transfer the private key part out of a smartcard to any other medium, nor to transfer only a public key part to a smartcard. If smartcards are used, a PIN pad can be connected to the computer where the csadm tool is running or to another computer.

This command is executed locally without any connection to a cHSM. Therefore the state or mode of any underlying cHSM is irrelevant for the command execution, and no authentication at any cHSM is necessary.

Syntax

For a public key:
`csadm [KeyType=<keytype>] PubKey=<source> SaveKey=<target>`

For a private key:
`csadm [KeyType=<keytype>] PrvKey=<source> SaveKey=<target>`

For a backup key:
`csadm [KeyType=<keytype>] BckKey=<source> SaveKey=<target>`

Parameter	Description
<keytype>	RSA (default) or EC
<source>	File name or key specifier where the key should be read from: For a public key: smartcard specifier or keyfile For private key: keyfile In case of an encrypted keyfile, the password has to be appended after the filename, separated by a '#'. If no password is given (or the password is <code>ask</code>), hidden password entry is performed. For backup key: Smartcard specifier where the two XOR parts of the backup key are read from two smartcards

Parameter	Description
<target>	Filename or key specifier of the target key For public key: keyfile For private key or backup key: smartcard specifier or keyfile In case of an encrypted keyfile, the password has to be appended after the filename, separated by a '#'. If the password is given as <code>ask</code> , hidden password entry is performed.

Example	<ul style="list-style-type: none"> Read an RSA public key from a smartcard and store in a keyfile <code>csadm PubKey=:cs2:cjo:USB0 SaveKey=C:\keys\pubkey1.key</code> Read an RSA backup key from two smartcards (two XOR parts) and store it in an encrypted keyfile <code>csadm BckKey=:cs2:cjo:USB0 SaveKey=C:\keys\RSAPubkey1_enc.key#ask</code>
----------------	--

Output	Upon successful execution of the command, no output is given.
---------------	---

4.7 Master Backup Key Commands

To provide backup functionality, a u.trust Anchor cHSM is able to store up to 256 Master Backup Keys to be used by various applications. Only AES keys (16, 24 or 32 bytes) are accepted as MBK. On every cHSM, an MBK named `AUTO-GEN` is autogenerated in slot 3. Further MBKs can be imported as needed.

An MBK can be generated on the u.trust Anchor cHSM and is stored split into key parts (key shares). The key parts of an MBK can be imported into the u.trust Anchor cHSM. If a new MBK should be imported into an MBK slot that already contains an MBK, the old MBK is overwritten with a new MBK without any verification except if the old MBK is the autogenerated MBK in MBK slot 3. In this case, the autogenerated MBK is automatically moved to another MBK slot and then the new MBK is imported into MBK slot 3.

This section describes how Master Backup Keys (MBK) can be managed remotely.



If you use an external database, make sure that you do not use the autogenerated MBK. This MBK is shown in the output of the `csadm MBKListKeys` command in the name column as `AUTO-GEN`. If an alarm or an external erase occurs, this MBK is deleted and the external database is inaccessible.

Use an MBK instead that you have generated, backed up in keyfiles or on smartcards (`csadm MBKGenerateKey` command) and imported into the u.trust Anchor cHSM (`csadm MBKImportKey` command). If now an alarm or an external erase occurs, reimport the same MBK from the keyfiles or smartcards.

If you access one external database from several devices, make sure that you use the same MBK in all these devices.

If you want to change the MBK an external database is encrypted with, perform an MBK rollover.



To unify the MBK storage of all applications of Utimaco IS GmbH (e.g. CXI), use the following convention:

MBK slot 1	Intermediate key (for example, while key is copied)
MBK slot 2	not used
MBK slot 3	AES key

We recommend using MBK slot 3 exclusively for any MBK storage (AES key). If the MBK shall be used to back up keys from the firmware module CXI, then the MBK in MBK slot 3 is mandatory: an MBK from any other MBK slot is not accepted by CXI.

The MBK slot number must not be confused with the PKCS#11 slot number.



To protect the key parts during transmission from/to the u.trust Anchor cHSM, they are encrypted with the session encryption key (AES) that was established when generating the secure messaging session with the u.trust Anchor cHSM. A secure messaging session is therefore necessary for remote MBK management.



Certain types of shell processes treat certain characters (commas, colons, semi-colons) differently. If the execution of a `csadm` command fails with an error message from the shell about an illegal parameter format, quoting parameter values may be necessary.

4.7.1 MBKCardCopy

This command copies the whole content of a MBK smartcard to another.

If records on the target smartcard already exist they will be overwritten without additional query. This command is executed locally without a connection to a device. Therefore, the state or mode of any underlying device is irrelevant for the command execution, and no authentication at any device is necessary.

Syntax	<code>csadm MBKCardCopy=<keyspec></code>
---------------	--

Parameter	Description
<code><keyspec></code>	Key specifier of the smartcard

Example	<code>csadm MBKCardCopy=:cs2:cjo:USB0</code>
----------------	--

Output	Upon successful execution of the command, no output is given.
---------------	---

4.7.2 MBKCardInfo

This command lists the contents of an MBK smartcard. Here, every MBK key share is stored in a separate record. The PIN pad can be connected to the host computer (USB port), where the csadm tool is running or to another computer.

This command is executed locally without a connection to a device. Therefore, the state or mode of any underlying device is irrelevant for the command execution, and no authentication at any device is necessary.

If you want to retrieve information not only about MBK shares but also about RSA key shares, ECDSA key shares, RSA keys and ECDSA keys stored on the smartcard, perform the `csadm GetCardInfo` command instead of performing the `csadm MBKCardInfo` command.

Syntax	<code>csadm MBKCardInfo=<cardspec></code>
---------------	---

Parameter	Description
<cardspec>	Smartcard specifier of the MBK smartcard

Example	csadm MBKCardInfo=:cs2:cjo:USB0
----------------	---------------------------------

Output	Upon successful execution of the command, the card information is returned. An MBK smartcard may contain up to 16 records/MBK parts. For each stored key share the following information is given:											
	REC	TYPE	ALG	LEN	NAME	TIMEGEN		K	ID	HASH		

	--											
	03	SHARE	AES	256	Test	13.04.2007 16:17:01		3	2	397620CEB7C61DBE		
	13	SHARE	AES	256	MbkAes28	09.05.2019 09:10:15		2	1	37BAFABF49C90D0C		
	14	XOR	AES	256	MbkAes29	09.05.2019 09:04:11		0	0	2722BDA0D7D6E821		
15	XOR	AES	256	MbkAes30	09.05.2019 08:57:24		0	0	D0779FB705960D8A			

```

13 SHARE AES 256 MbKaes28 09.05.2019 09:10:15 2 1 37BAFABF49C90D0C
14 XOR AES 256 MbKaes29 09.05.2019 09:04:11 0 0 2722BDA0D7D6E821
15 XOR AES 256 MbKaes30 09.05.2019 08:57:24 0 0 D0779FB705960D8A

```

Parameter	Description
REC	Record number
TYPE	Share type XORB : XOR half SHARE : Key share
ALG	MBK algorithm
LEN	Key length in bits
NAME	Name of an MBK or <NoName> if no key name is given
TIMEGEN	Date and time of key generation
M	Number of shares that are necessary to recombine MBK Only relevant if TYPE is SHARE
ID	ID of the share
HASH	Check value over MBK (usually: first 8 bytes of ISO-hash MDC2 over MBK; if ID = 0 or 1 : first respectively second 8 bytes of ISO-hash MDC2 over MBK).

Table 13: Meaning of the output parameters of csadm MBKCardInfo

4.7.3 MBKPINChange

This command changes the PIN on an MBK smartcard.

The PIN pad has to be connected to a USB port of that computer where the csadm tool is running. Watch the PIN pad's display: Enter the old PIN first, enter the new PIN then and confirm the new PIN. This command is executed locally without a connection to a cHSM. Therefore, the state or mode of any underlying cHSM is irrelevant for the command execution, and no authentication at any cHSM is necessary.

This command changes the PIN on a smartcard where an MBK is stored.

This command is executed locally without a connection to a device. Therefore, the state or mode of any underlying device is irrelevant for the command execution, and no authentication at any device is necessary.

Syntax	<code>csadm MBKPINChange=<cardspec></code>
---------------	--

Parameter	Description
<code><cardspec></code>	Smartcard specifier of the MBK smartcard

Example	<code>csadm MBKPINChange=:cs2:cjo:USB0</code>
----------------	---

Output	Upon successful execution of the command, no output is given.
---------------	---



The PIN pad can be connected to the computer where the csadm tool is running (USB port) or to another computer.

Please follow the instructions on the display of the PIN pad.

4.7.4 MBKImportKey

This command imports the key shares of an AES MBK from two or more keyfiles.



The usage of encrypted keyfiles and hidden password entry is strongly recommended.



Before you perform this command, verify which backup files have been generated because they might become inaccessible after the import. After performing the `csadm MBKImportKey` command, regenerate these backup files.

Syntax	<code>csadm [Dev=<device>] <Authentication> Key=<keyspec> MBKImportKey=<slot_no></code>
Authentication	Permission level 2 in user group 6. Additionally the command has to be executed within a secure messaging session.
Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<keyspec>	Key specifier where the new MBK should be loaded from <file#password> If password is given as <code>ask</code> , a secure password entry will be performed. No password is needed in case of a clear-text keyfile.
<slot_no>	Slot number on the u.trust Anchor cHSM to which the MBK should be imported. It must be MBK slot 3 for AES keys. If there is already an old MBK in the MBK slot indicated by <code>slot_no</code> , this old MBK is overwritten by the new MBK to be imported. The only exception is the case that <code>slot_no</code> is 3 and the old MBK is an autogenerated MBK (shown as the MBK name <code>AUTO-GEN</code> in the output of the <code>csadm MBKListKeys</code> command). In this case, the autogenerated MBK is moved from MBK slot 3 to MBK slot 7 (or to the next free MBK slot > 7 if MBK slot 7 is already occupied by another MBK) and the new MBK is imported into MBK slot 3. To verify which MBK is in a certain MBK slot, use the <code>csadm MBKListKeys</code> command. Consider that it is important which MBK is in use in MBK slot 3 by the device because this MBK is used by the backup commands to protect the backup files that are generated by these commands. It is important to note down which MBK has been used for these backup commands because for a successful restoring of these backup files at a later date it is necessary that the same MBK is in MBK slot 3. Otherwise, for example, after the execution of a <code>csadm MBKImportKey</code> command or after an MBK rollover, the backup files are inaccessible.
Example	<code>csadm LogonPass=Paul,<Paul key file> ask Key=mbk1.key#swordfish,mbk2.key#sesame MBKImportKey=3</code>

Output

Upon successful execution of the command, no output is given.

4.7.5 MBKGenerateKey

This command generates an AES MBK on the cHSM, splits it into `n` key shares and transmits the encrypted key shares (encrypted with the session key of the current Secure Messaging session) to the host. The key shares will be decrypted on the host and stored either on `n` smartcards or in `n` key files. The newly generated MBK is not stored on the cHSM. To store the MBK on the cHSM it has to be re-imported, see command `csadm MBKImportKey`.

If smartcards are used (recommended), a PIN pad including smartcard reader has to be connected to the computer where the csadm tool is running. Watch the PIN pad's display for instructions on further command processing.

Syntax

```
csadm [Dev=<device>] <Authentication> Key=<keyspec>
MBKGenerateKey=<keytype>,<keylen>[,<n>,<m>,<keyname>]
```

Authentication

Permission level 2 in user group 6.
Additionally the command has to be executed within a secure messaging session.

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<keyspec>	<ul style="list-style-type: none"> Key specifier where the generated MBK shares should be stored <code><smartcard,record number></code>, e.g. <code>:cs2:cjo:USB0,15</code> The default record number for AES keys is 15. List of filenames and (if the key should be password protected) passwords, <code><file#password></code> (for example, <code>file1#pwd1,file2#pwd2</code>). If the password is given as the string <code>ask</code>, hidden password entry is performed.
<keytype>	The key type of the MBK to be generated, must be 'AES'
<keylen>	Key size of the MBK to be generated
<n>	Number of key shares to be generated Default: 2
<m>	Number of key shares required to recombine key Default: 2

Parameter	Description
<keyname>	Key name, up to 8 characters with ANSI / ISO-8859-1 encoding. Do not name the MBK to be created <code>AUTO-GEN</code> because this is the name of the autogenerated MBK.

Example	<ul style="list-style-type: none"> ▪ <code>csadm LogonSign=ADMIN,:cs2:cjo:USB0 Key=:cs2:cjo:USB0,1 MBKGenerateKey=AES,24,2,2,MyAESKey</code> ▪ <code>csadm LogonPass=Paul,mypassword Key=mbk1.key#swordfish,mbk2.key#sesame MBKGenerateKey=AES,32,2,2,MyAESKey</code> ▪ <code>csadm LogonPass=Paul,ask Key=:cs2:cjo:USB0,15 MBKGenerateKey=AES,32,5,3,AES_new</code>
----------------	---

Output	Upon successful execution of the command, no output is given.
---------------	---

- If smartcards are used (recommended):
The PIN pad can be connected to the computer where the csadm tool is running (USB port) or to another computer.
Watch the display of the PIN pad for instructions on further command processing.
- If keyfiles are used:
We strongly recommend using encrypted keyfiles.



The newly generated MBK is not stored on the device. To store the MBK on the device it has to be imported with the command `MBKImportKey`.



To protect the key parts during transmission from/to the device, they are encrypted with the session key (AES) that was exchanged on establishment of the secure messaging session with the device.

A secure messaging session is therefore necessary for remote MBK management.

4.7.6 2020-0037 MBKListKeys

This command lists all MBKs currently stored on a u.trust Anchor cHSM.

Syntax	csadm <Authentication> <Session> MBKListKeys
---------------	--

Authentication	Permission Level 2 in user group 6 (02000000)
-----------------------	---

Example	csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,:cs2:cjo:USB0 MBKListKeys
----------------	---

Output	<p>Upon successful execution, the command returns a list of all MBKs currently stored on the u.trust Anchor CHSM.</p> <pre> slot name len algo type k generation date key check value ----- 3 <NoName> 32 AES XOR 2 07/08/06 10:35:50 106B5E4E84031BDE </pre>
---------------	---

Parameter	Description
slot	The slot the MBK is stored in
name	The name of the MBK, or <NoName> if no key name is given. AUTO-GEN indicates an autogenerated MBK.
len	The size of the MBK in byte
algo	The algorithm of the MBK, is always AES
type	The type of the MBK XOR: The MBK was exported in two XOR halves Share: The MBK was exported more than two key shares
generation date	The date and time of the generation of the MBK
key check value	The key check value used to identify the MBK, ISO-hash MDC2 over MBK

Table 14: Meaning of the return parameters of csadm MBKListKeys

4.7.7 2020-0037 MBKCopyKey

This command copies one key share of a MBK from one storage location to another. This command is executed locally without a connection to a CHSM. Therefore, the state or mode of any underlying CHSM is irrelevant for the command execution, and no authentication at any CHSM is necessary.

With this command the storage location of an MBK can be migrated from a smartcard to a keyfile or vice versa.

If smartcards are used: A PIN pad including a smartcard reader can be connected to the computer, where the csadm tool is running (USB port) or to another computer. Watch the display of the PIN pad for instructions on further command processing.

If keyfiles are used: We strongly recommend using encrypted keyfiles. If encrypted keyfiles are used, we strongly recommend using a hidden password entry.

Syntax	<code>csadm Key=<keyspec_source> MBKCopyKey=<keyspec_target></code>
---------------	---

Parameter	Description
<code><keyspec_source></code>	Key specifier indicating where the key share should be loaded from
<code><keyspec></code>	<p>Key specifier indicating where the generated MBK shares should be stored <code><smartcard,record number></code>, e.g. <code>:cs2:cjo:USB0,15</code></p> <p>The default record number for AES keys is 15.</p> <p>List of filenames and (if the key should be password protected) passwords <code><file#password></code> (for example, <code>file1#pwd1,file2#pwd2</code>).</p> <p>If password is given as the string <code>ask</code>, hidden password entry is performed. If no password is given, the key file is stored in plaintext.</p>
<code><keyspec_target></code>	Key specifier indicating where the key share should be stored

Example	<ul style="list-style-type: none"> <code>csadm Key= mbk1.key#swordfish MBKCopyKey=:cs2:cjo:USB0,15</code> <code>csadm Key=:cs2:cjo:USB0,15 MBKCopyKey=mbk2.key#ask</code>
----------------	---

Output	Upon successful execution of the command, no output is given.
---------------	---

4.8 Commands for Administration of u.trust Anchor LAN

This command group explicitly addresses the u.trust Anchor LAN appliance itself; commands will not be forwarded to the mounted u.trust Anchor PCIe card. Instead, they will be processed by the control module of the TCP-Server (csxlan-daemon) and responded to in the same way a firmware module would respond to a command.

These commands are used to administrate a u.trust Anchor LAN remotely (for example, to get/set its configuration).



Since the commands of this group are not directed to any u.trust Anchor PCIe card but only to the u.trust Anchor LAN, the presence, mode or state of eventually underlying devices are not relevant for their performance. For the same reason, none of these commands have to be authenticated using the u.trust Anchor's authentication mechanism.

Nevertheless, some commands of this group are protected against unauthorized use with an own authentication mechanism: these commands have to be authenticated with the root password of the u.trust Anchor LAN. See also *u.trust anchor LAN V5 – Administration Manual*.



Certain types of shell processes treat certain characters (commas, colons, semi-colons) differently. If the execution of a csadm command fails with an error message from the shell about an illegal parameter format, quoting parameter values may be necessary.

4.8.1 CSLGetConnections



The `csadm CSLGetConnections` command is only performed on the u.trust Anchor LAN device.

This command lists all current connections to the LAN device.

Syntax	<code>csadm [Dev=<device>] CSLGetConnections</code>
Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
Example	<code>csadm Dev=10.17.1.9 CSLGetConnections</code>

Output	Upon successful execution of the command, a list of all current LAN connections is returned. current connections to CryptoServer LAN
	<pre># prot port address ----- 1 TCP 1131 193.168.4.122 2 TCP 2563 194.169.4.098 3 TCP 1137 195.170.4.122</pre>

Parameter	Description
#	Connection number
prot	Used protocol (either UDP or TCP)
port	Port on the administration computer used to open the connection
address	IP address of the administration computer

Table 15: Meaning of the output parameters of csadm CSLGetConnections



Up to 10,000 connections can be established to one LAN device simultaneously.

4.8.2 CSLGetVersion



The `csadm CSLGetVersion` command is only performed on the u.trust Anchor LAN device.

This command displays the version number of the LAN device.

Syntax	<code>csadm [Dev=<device>] CSLGetVersion</code>
---------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example

```
csadm Dev=10.17.1.9 CSLGetVersion
```

Output

Upon successful execution of the command, the version number of the LAN device is returned.

```
CSLAN      4.1.0
```

4.8.3 CSLGetStatus



The `csadm CSLGetStatus` command is only performed on the u.trust Anchor LAN device.

The `CSLGetStatus` command displays status information about the LAN device.



This command is only implemented for LAN device with version 3.3.0 and later. For older versions of the LAN device, the output of this status information is not supported.

Most of the information is only available for CryptoServer LAN V4 and later.

Syntax

```
csadm [Dev=<device>] CSLGetStatus
```

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example

```
csadm Dev=10.17.1.9 CSLGetStatus
```

Output

Upon successful execution of the command, the status information of the LAN device is returned.

<i>Parameter</i>	<i>Description</i>
uptime	System uptime (days, hours)
fan speed [rpm]	<p>Fan speed for all fans as rotation per minute (rpm). For a CSLAN V4b, three values are shown. The first value is for the CPU fan, the second is for the first internal fan, and the third is for the second internal fan. For a CSLAN V4c, two values are shown. The first value is for the first internal fan, and the second is for the second internal fan. This status information is only available for CryptoServer LAN V4 and later. A value of 0 for the fan speed indicates a broken fan. In this case, create an RMA (Return Merchandise Authorization) according to the Chapter "Contact Address for Support Queries".</p>
CPU temperature [C]	<p>CPU temperature for CPU of the CryptoServer LAN motherboard in degree Celsius. This status information is only available for CryptoServer LAN V4 and later.</p>
redundant power supply	<ul style="list-style-type: none"> Status of the redundant power supply units which can be OK or FAILED. This status information is only available for CryptoServer LAN V4 and later

Table 16: Meaning of the output parameters of csadm CSLGetStatus

4.8.4 CSLGetLogFile



The `csadm CSLGetLogFile` command is only performed on the u.trust Anchor LAN device.

This command displays the log file of the LAN device.

Syntax	<code>csadm [Dev=<device>] CSLGetLogFile[=<fileno>]</code>
---------------	--

<i>Parameter</i>	<i>Description</i>
<device>	<p>Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.</p>
<fileno>	<p>Number of the logfile on the LAN device that shall be from 0 to 9. If omitted, the first logfile (number 0, <code>/var/log/csxlan.log</code>) is retrieved. If the number is invalid, the error code 0xB9067005 is returned</p>

Example

```
csadm Dev=10.17.1.9 CSLGetLogFile
```

Output

Upon successful execution of the command, the content of the log file is returned.



Use '>' to dump the output into a file, for example,

```
csadm GetConfigFile > c:\temp\csxlan.conf
```

The log file is formatted according to the UNIX style.



The content of the log file depends on the trace level setting of the LAN device. See the respective Administration Manual for details.

4.8.5 CSLGetConfigFile



The `csadm CSLGetConfigFile` command is only performed on the u.trust Anchor LAN device.

This command displays the content of the configuration file `/etc/csxlan.conf` of the LAN device.

Syntax

```
csadm [Dev=<device>] CSLGetConfigFile
```

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example

```
csadm Dev=10.17.1.9 CSLGetConfigFile
```

Output

Upon successful execution of the command, the content of the configuration file is returned.



Use '>' to dump the output into a file, for example,
`csadm GetConfigFile > c:\temp\csxlan.conf`

The configuration file is formatted according to the UNIX style.

4.8.6 CSLSetTracelevel



The `csadm CSLSetTraceLevel` command is only performed on the u.trust Anchor LAN device.

This command sets the trace level (log level) of a LAN device. Depending on the trace level, more or less information is written into the log file `/var/log/csxlan.log`.



The new trace level is only a temporary setting. The next time LAN device is (re)started, the level specified by the `LogLevel` variable (CSLANOS version $\geq 5.0.0$) in the `/etc/csxlan.conf` configuration file is used again.

The following table shows the values that are supported for CSLANOS version $\geq 5.0.0$.

Value	Trace level	Description
0x03	Error	Error messages. The values 0x00, 0x01 and 0x02 have the same effect.
0x04	Warning	Warning messages
0x05	Notice	Normal but significant condition
0x06	Info	Informational messages
0x07	Debug	Debug level messages

Table 17: Trace levels for CSLANOS version $\geq 5.0.0$

The `- 0x01` value is not supported by the `csadm CSLSetTraceLevel` command although the corresponding `OFF` value of the `LogLevel` variable in the `/etc/csxlan.conf` file is supported.

Syntax	<code>csadm [Dev=<device>] CSLSetTraceLevel=<password>,<level></code>
---------------	---

Authentication	Root password of the LAN device, see parameter <code><password></code>
-----------------------	--

Parameter	Description
<code><device></code>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<code><password></code>	Root password of the LAN device <ul style="list-style-type: none"> ▪ for hidden password entry use the string ask ▪ We strongly recommend using a hidden password entry. ▪ root password of the LAN device in clear text
<code><level></code>	Desired trace level as stated in the table above, for example, <code>0x06</code> .

Example	<code>csadm Dev=192.168.4.1 CSLSetTracelevel=123456,0x06</code>
----------------	---

Output	Upon successful execution of the command, no output is returned.
---------------	--

4.8.7 CSLShutdown



The `csadm CSLShutdown` command is only performed on the u.trust Anchor LAN device.

This command shuts down the LAN device as the Linux command `shutdown -h` would do at the local console. All programs and services are stopped, all devices are unmounted and the system is put into `RunLevel 0`. The LAN device can then be powered off.

Syntax	<code>csadm [Dev=<device>] CSLShutdown=<password></code>
---------------	--

Authentication	Root password of the LAN device, see parameter <code><password></code>
-----------------------	--

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<password>	Root password of the LAN device <ul style="list-style-type: none"> ▪ for hidden password entry use the string ask ▪ We strongly recommend using a hidden password entry. ▪ root password of the LAN device in clear text

Example	csadm Dev=192.168.4.1 CSLShutdown
----------------	-----------------------------------

Output	Upon successful execution of the command, no output is returned.
---------------	--

4.8.8 CSLReboot



The `csadm CSLReboot` command is only performed on the u.trust Anchor LAN device.

This command reboots the LAN device the same way the Linux command `shutdown -r` would do at the local console. All programs and services are stopped, all devices are unmounted and the system is rebooted again.

Syntax	csadm [Dev=<device>] CSLReboot=<password>
---------------	---

Authentication	Root password of the LAN device, see parameter <password>
-----------------------	---

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<password>	Root password of the LAN device <ul style="list-style-type: none"> ▪ for hidden password entry use the string ask ▪ We strongly recommend using a hidden password entry. ▪ root password of the LAN device in clear text

Example

```
csadm Dev=192.168.4.1 CSLReboot
```

Output

Upon successful execution of the command, the device reboots.

4.8.9 CSLGetTime



The `csadm CSLGetTime` command is only performed on the u.trust Anchor LAN device.

This command reads the local system time and UTC time of the LAN device (not the clock of the PCIe card) and outputs the date and time.

To get the time of the PCIe card, perform the `csadm GetTime` command instead.

Syntax

```
csadm [Dev=<device>] CSLGetTime
```

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example

```
csadm Dev=192.168.4.1 CSLGetTime
```

Output

Upon successful execution of the command, the date and time of the LAN device are returned.
The first line shows the time of the LAN device in the time zone of the LAN device (local time).
The second line shows the same time converted into the GMT (UTC) time zone (UTC time). This is not necessarily the time of the device (PCIe card).
The date is shown in the DD:MM.YYYY format.

```
date: 03.03.2015    time: 10:58:50 (local time)
date: 03.03.2015    time: 10:58:50 (UTC time)
```


4.8.10 CSLSetTime



The `csadm CSLSetTime` command is only performed on the u.trust Anchor LAN device.

This command sets the local system time of the LAN device (not the clock of the PCIe card).

To set the time on the PCIe card, perform the `csadm SetTime` command instead.



The LAN device retrieves its time from a time source. This time source might be an NTP server. As of CSLANOS 5.1, a PCIe clock card is supported as a time source as well.

If you set the time on the LAN device manually by performing the `csadm CSLSetTime` command and if this causes the time difference between the time on the LAN device and the time of the time source to be larger than 1000 s, this causes an error message and the NTP daemon is terminated automatically. In such case, the time on the device will not be set.

Syntax

```
csadm [Dev=<device>] CSLSetTime=<password>,<time>
```

Authentication

Root password of the LAN device, see parameter `<password>`

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<password>	Root password of the LAN device: <ul style="list-style-type: none"> For hidden password entry use the string ask We strongly recommend using a hidden password entry. root password of the LAN device in clear text
<time>	<ul style="list-style-type: none"> <code>YYYYMMDDhhmmss</code> The time is set manually by using the format sample YYYYMMDDhhmmss YYYY=year, MM=month, DD=day, hh=hour, mm=minute, ss=second <code>SYSTEM</code> The system time of the administration computer is used

Example	<code>csadm Dev=192.168.1.1 CSLSetTime=123456,SYSTEM</code>
----------------	---

Output	<p>Upon successful execution of the command, the date and time of the LAN device are returned.</p> <p>The second line shows the time of the LAN device in the time zone of the LAN device (local time).</p> <p>The third line shows the same time converted into the GMT (UTC) time zone (UTC time). This is not necessarily the time of the device (PCIe card).</p> <p>The date is shown in the DD:MM.YYYY format.</p> <pre>Time successfully set to: date: 06.12.2019 time: 11:16:53 (local time) date: 06.12.2019 time: 10:16:53 (UTC time)</pre>
---------------	--

4.8.11 CSLGetSerial



The `csadm CSLGetSerial` command is only performed on the u.trust Anchor LAN device.

This command reads and outputs the serial number of the LAN device appliance (not the serial number of the mounted PCIe card).

Syntax	<code>csadm [Dev=<device>] CSLGetSerial</code>
---------------	--

Parameter	Description
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	<code>csadm Dev=192.168.1.1 CSLGetSerial</code>
----------------	---

Output	Upon successful execution of the command, the serial number of the LAN device is returned.
---------------	--

4.8.12 CSLGetLoad



The `csadm CSLGetLoad` command is only performed on the u.trust Anchor LAN device.

This command reads and outputs the workload of the PCIe card in percent.

Syntax	<code>csadm [Dev=<device>] CSLGetLoad</code>
---------------	--

Parameter	Description
<code><device></code>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.

Example	<code>csadm Dev=192.168.1.1 CSLGetLoad</code>
----------------	---

Output	Upon successful execution of the command, the work load of the device (ratio of the time that requests/commands spend on the PCIe card to the total time) is returned.
---------------	--



The information of the work load of the device is read from the display module of the LAN device. If the display module is busy (i.e. an operator is working at the display) this information is not available and the command does not output any value.



The value returned by the command is not the work load at that time. It is an average value of the last 64 measurements. The value is continuously recalculated. If no packets arrive, it is recalculated once per second. If more packets arrive, it is recalculated more often, up to once per every few milliseconds.

5 Basic Administration Tasks

5.1 Entering and Leaving the 'Administration only' Mode

The device can switch temporarily between the normal Operational Mode and the restricted *Operational Mode – Administration-Only* without being restarted. In *Operational Mode – Administration-Only*, only functions needed for the device administration are available, and all cryptographic functions are blocked.

The device can switch between the two modes with the `csadm SetAdminMode` command. The operating mode set with this command is only relevant until the next time the device is restarted.

The device can be in *Operational mode – Administration-Only* which allows the use of only administrative functions. All cryptographic services are blocked. To switch a device from *Operational Mode – Administration-Only* to *Operational Mode* and activate the cryptographic services, perform these steps.

Prerequisites

- The device is in state `INITIALIZED`, as well as in *Operational Mode – Administration-Only*.
- The authentication token(s) of a user (or a group of users) with at least user permission 22000000 is available. If the device is set up for the first time, this means that the authentication key of the default administrator ADMIN is at hand, provided by the u.trust Anchor Global Administrator.

Procedure

1. Perform the `csadm GetState` command.
2. Perform the `csadm SetAdminMode` command.
Example: `csadm Dev=4001@192.168.1.1 LogonSign=adminUsr,:cs2:cjo:USB0 SetAdminMode=0`
3. Perform the `csadm GetState` command again to verify the mode of the device.



The device has successfully been set back to *Operational Mode*.

5.2 Generating a User Authentication Key



On FIPS cHSMs, only RSA signatures are supported.

For every user who shall authenticate themselves towards the device with an RSA or ECDSA signature, a user authentication key has to be generated. Basically, the private part of the user authentication key can either be stored in a keyfile, which optionally is password-protected, or on a smartcard. As the private part of the key cannot be read out from the smartcard, the usage of smartcards is more secure and therefore recommended.



Before delivery, the Default Administrator ADMIN is created on every device as the default administrator. The authentication token of this user, labelled "ADMIN," is shipped as clear text keyfile (`ADMIN.key`) and is initially stored on every smartcard that is shipped by Utimaco IS GmbH (with default PIN 123456).

The customer should either replace the authentication token of the ADMIN with a self-generated RSA key (see command `ChangeUser`) or create other users with sufficient permissions on the device and then delete the Default Administrator ADMIN.

Prerequisites

- If smartcards shall be used, at least one smartcard per user has to be at hand. If the smartcard already contains a key (RSA-Key or ECC-Key), this key will be overwritten.
- A PIN pad has to be connected to the USB port of the computer where csadm is installed or to another computer



We strongly recommend the generation of the new authentication token directly on a trustworthy USB flash drive, that the generated keyfile is copied on as many smartcards (provided by Utimaco) as required by the security policy of your company, and the USB flash drive and the smartcards are stored in a secure safe.

Procedure

1. Create a new RSA or ECDSA key pair as an encrypted keyfile by using the csadm command `GenKey` .

```
csadm NewPassword=ask KeyType=RSA GenKey=E:\myKeys\myRSA.key,2048,Admin1
generating RSA key: E:\myKeys\myRSA.key, 2048 bits, owner: Admin1
Enter New Passphrase:
Repeat New Passphrase:
```



The generated keyfile can already be used to authenticate towards the device.
If the key shall not be copied on a smartcard, the following steps can be skipped.

2. Store the key on a smartcard by using the command `SaveKey`.

Example:

```
csadm KeyType=RSA PrvKey=E:\myKeys\myRSA.key#ask SaveKey=:cs2:cjo:USB0
Enter Passphrase:
```

3. Follow the instructions on the PIN pad.
4. Check that the key is stored on the smartcard by using the csadm command `GetCardInfo`.

Example:

```
csadm GetCardInfo=:cs2:cjo:USB0
```

5. Follow the instructions on the PIN pad display.
6. Change the PIN of the smartcard by using the csadm command `ChangePIN`.

Example:

```
csadm ChangePIN=:cs2:cjo:USB0
```

7. Follow the instructions on the PIN pad display.
Depending on your security policy, the following step can be omitted, if steps 2 to 7 will be performed several times. The additional smartcards that will be created that way, have to be used as a backup for the generated key.
8. Create a backup of the private key on two smartcards with the command `BackupKey`.
A backup smartcard is used for key storage only. It cannot be used for authentication

towards the device. A backup smartcard contains only one XOR-half of a key, so two backup cards are necessary to regain the complete key. The key halves can be read out of the smartcard to generate new administration smartcards containing the stored key at a later time with the command `SaveKey`.

Example:

```
csadm KeyType=RSA PrvKey=E:\myKeys\MyRSA.key#ask BackupKey=:cs2:cjo:USB0
```

9. Follow the instructions on the PIN pad.
 10. Change the PIN of the backup smartcard by using the `csadm` command `ChangePIN`.
- Example:

```
csadm ChangePIN=:cs2:cjo:USB0
```

11. Follow the instructions on the PIN pad display.
12. Store the keyfile at a protected place (for example, on a USB flash drive in a safe) or delete it (in case you made a backup copy on minimum two smartcards).



A user authentication key has been generated successfully.

The new user authentication key can now be assigned to every new device user on creation (see the `AddUser` command) who shall use RSA or ECDSA signature authentication. Furthermore, the key can be assigned to existing device users using RSA or ECDSA signature authentication with the `ChangeUser` command.

The PIN pad (if needed) has to be connected to the computer where the `csadm` tool is running (USB port) or to another computer.

If the authentication of the command `AddUser` requires a smartcard and the source of the new user's public key is a smartcard too, follow the instructions on the display of the PIN pad. You'll have to insert the smartcard for the command authentication (old key) first, and then the smartcard containing the user's new authentication key.

6 Advanced Administration Tasks

6.1 Generating and Verifying Signed Audit Log Files

When generating and using a signed audit log, an audit log signature key is required. This key can be generated or retrieved only if the device is in *Operational Mode*. Therefore, first verify whether the device is in this mode.

Checking the Operation Mode of the cHSM

1. Perform the `csadm GetState` command according to the following example.

```
csadm Dev=4001@192.168.1.1 GetState | grep mode
```

The output should be as follows:

```
mode = Operational Mode
```

Generating an Audit Log Signature Key

The audit log signature key is stored in the `auditkey.db` database. This database contains only one audit log signature key and nothing else.

1. Verify whether the database exists on your device by performing a command according to the following example:

```
csadm Dev=4001@192.168.1.1 ListFiles | grep auditkey.db
```

If the output of this command starts with `FLASH\auditkey.db`, this database exists.
Example output:

```
FLASH\auditkey.db 1785 -
```



If the `auditkey.db` database is available, the generation of the key can be skipped and the verification can be started.

2. Generate an audit log signature key by performing the `csadm GenerateAuditLogKey` command according to the following example.

```
csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,:cs2:cjo:USB0  
GenerateAuditLogKey=1 > ./pub_audit_log_key.txt
```

It is important that the output of the `csadm GenerateAuditLogKey` command is redirected into a file as shown in the example because this file is used later on to generate signed audit log files.



The Audit Log Signature Key has been successfully created and is stored in the `auditkey.db` database.

Generating and Verifying Signed Audit Log Files

1. Verify which unsigned audit log files exist on the device.

```
csadm Dev=4001@192.168.1.1 ListFiles | grep .log
```

Example output:

```
FLASH\audit13AC97.log 155713 -
```

Consider that the unsigned audit log files are stored in the FLASH memory of the device and that short or long audit log filenames are used for them. In contrast, signed audit log files are always generated on the computer `csadm` is running on and extra long audit file names are used for these audit log files.

2. To generate signed audit log files on the computer `csadm` is running on, perform the `csadm GetSignedAuditLog` command according to the following example. Assign the redirected output of the `csadm GenerateAuditLogKey` command or the `csadm GetAuditLogKey` command (for example, the `pub_audit_log_key.txt` file) to the `AuditPubKey` parameter.

```
csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,:cs2:cjo:USB0 AuditPubKey=./
pub_audit_log_key.txt GetSignedAuditLog=./logs
```

This example generates signed audit log file(s) in the logs directory.

3. To verify a signed audit log file, perform a command according to the following example.

```
csadm Dev=4001@192.168.1.1 AuditPubKey=./pub_audit_log_key.txt
VerifySignedAuditLog=./logs/CS123456_0013AC97.log,print
```



The Signed Audit Log Files have been successfully verified,

Deleting Unsigned Audit Log Files

If you want to delete the unsigned audit log files in the FLASH memory of the device that are related to the signed audit log files on the computer csadm is running on at a later date, perform the `csadm ClearAuditLogFiles` command according to the following example.

```
csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,:cs2:cjo:USB0
ClearAuditLogFiles=3a,./logs
```

6.2 Using the Signed Configuration File `cmds.scf`

The signed configuration file `cmds.scf` can be used to disable some specific external firmware interfaces or/and to set specific permission requirements (higher than the permissions required by default) for some external cHSM functions.

This section describes this by means of examples how to use the file `cmds.scf`.

Preconditions

- The cHSM is a FIPS cHSM, is operating in *Operational Mode*, and is not in an ERROR state.
- You have received from Utimaco a signed configuration file `cmds.cfg` containing the required restrictions defined by you.

- The version of the csadm tool installed on your host computer is at least 1.8.11b.

Procedure

1. Load the signed configuration file into the CryptoServer.

Example:

```
csadm Dev=4001@192.168.1.1 LogonSign=ADMIN,D:\keys\ADMIN.key#ask LoadFile=N:\User\Utimaco\cmds.scf
```

2. Check that the signed configuration file `cmds.scf` has been successfully loaded into the cHSM.

Example:

```
csadm Dev=4001@192.168.1.1 ListFiles
```

3. Restart the cHSM.

Example:

```
csadm Dev=4001@192.168.1.1 Restart
```

4. Check the corresponding log file entries to ensure the signed configuration file has been loaded and the provided settings are used.

- a. Check the boot log file to ensure the settings configured in section `[DisableSFC]` are used.

Example:

```
csadm Dev=4001@192.168.1.1 GetBootLog
```

You should see an entry for the functions that have been disabled.

Example:

```
CMDS/DSOM: 0x083 - disabled 0 1
```

Check that the functions you wanted to be disabled are blocked.

Example:

```
csadm Dev=4001@192.168.1.1 cmd=0x083,0,1
```

You should see the following error message:

```
Error B0830061
CryptoServer module CMDS, Command scheduler
This function is not available in this HSM configuration
```

- b. Check the audit log file to ensure the settings configured in section [Permissions] are used. By default, the csadm command GetAuditLog does not have to be authenticated.

Example:

```
csadm Dev=4001@192.168.1.1 GetAuditLog
```

With the appropriate authentication, you should see the following audit log entries for the `cmds.cfg` used as an example in the preconditions.

```
csadm Dev=4001@192.168.1.1 LogonPass=demo,ask
GetAuditLog

19.02.16 14:10:43 SMOS Ver. 3.3.4.1
successfully started
19.02.16 14:10:44 FC:0x083 SFC:5
Configured Permission=FF000000
19.02.16 14:10:44 FC:0x083 SFC:6
Configured Permission=FF000000
19.02.16 14:10:44 FC:0x083 SFC:12
Configured Permission=FF000000
19.02.16 14:10:44 FC:0x087 SFC:10
Configured Permission=00200000
19.02.16 14:10:44 FC:0x068 SFC:17
Configured Permission=6F000000
```

7 Contact Address for Support Queries

You can reach us from Monday to Friday, 09.00 a.m. to 05.00 p.m., Central European Time (CET).

Utimaco IS GmbH
Germanusstr. 4
52080 Aachen
Germany

RMA Query

If you need to send the device back to Utimaco IS GmbH, please open a new RMA case (Return Merchandise Authorization). We request that you use the following web address. RMA cases cannot be opened by email or phone.

<https://support.hsm.utimaco.com/support/rma/new>

Other Support Queries

- Mail (preferred contact method)
support@utimaco.com
Attach the diagnostic information to your email.
- Web portal
<https://support.hsm.utimaco.com/support/cases/new/>
The diagnostic information will be requested in our response if necessary.
- By phone
AMERICAS +1-844-UTIMACO (+1 844-884-6226)
EMEA +49 800-627-3081
APAC +81 800-919-1301
The diagnostic information will be requested in our response if necessary.