

# u.trust Anchor

## Migration Guide

## 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	<a href="https://support.hsm.utimaco.com/">https://support.hsm.utimaco.com/</a> <a href="mailto:support@utimaco.com">support@utimaco.com</a>
Document Version	1.0.8
Product Version	6.0.0
Date	2024-10-24
Document No.	2023-0012
Status	<b>PUBLISHED</b>

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

<b>1</b>	<b>Introduction .....</b>	<b>5</b>
1.1	About this Guide .....	5
1.1.1	Target Audience .....	5
1.1.2	Document Conventions .....	5
1.1.3	Abbreviations .....	6
<b>2</b>	<b>About the u.trust Anchor Se.....</b>	<b>8</b>
2.1	Overview .....	8
2.1.1	Hardware .....	8
2.1.1.1	Temperature-dependent Behavior of the u.trust Anchor Se Device .....	9
2.1.2	Software .....	10
2.1.2.1	Boot Process .....	11
2.1.2.2	Operating System - SMOS Façade.....	11
2.1.2.3	Firmware Modules .....	11
2.1.3	Administration Tools .....	13
2.1.3.1	CryptoServer Command-Line Administration Tool (csadm) .....	14
2.1.3.2	gladm Administration Tool .....	16
2.1.3.3	p11tool2 Administration Tool.....	20
<b>3</b>	<b>Backup and Restore .....</b>	<b>22</b>
3.1	Preconditions .....	22
3.2	Backing up Databases on CryptoServer SeGen2 .....	25
3.2.1	Generate Master Backup Key (MBK) .....	25
3.2.2	Backup User Database .....	27
3.2.2.1	Backup User with csadm BackupDatabase .....	27
3.2.2.2	Backup with csadm BackupUser .....	29
3.2.3	Backup Key Database.....	30
3.2.3.1	Backup Key with csadm BackupDatabase.....	30
3.2.3.2	Backup Key with cxitool Backupkey .....	32
3.2.3.3	Backup Key with p11tool2 BackupInternalKeys .....	35
3.3	Restoring Databases on u.trust Anchor Se.....	36
3.3.1	Import Master Backup Key (MBK) .....	36
3.3.2	Restore User Database .....	38
3.3.2.1	Restore User with csadm RestoreDatabase .....	38
3.3.2.2	Restore User with csadm RestoreUser .....	39

---

3.3.3	Restore Key Database .....	44
3.3.3.1	Restore Key with csadm RestoreDatabase.....	44
3.3.3.2	Restore Key with cxitool RestoreKey.....	46
3.3.3.3	Restore Key with p11tool2 RestoreInternalKeys.....	48
4	<b>References .....</b>	<b>50</b>
5	<b>Contact Address for Support Queries .....</b>	<b>51</b>

# 1 Introduction

Thank you for purchasing our u.trust Anchor security system. We hope you are satisfied with our product. Please do not hesitate to contact us if you have any questions or comments.

## 1.1 About this Guide

This guide provides information about the fundamentals of Utimaco's hardware security module u.trust Anchor Se, its security mechanisms and explains the migration process between u.trust Anchor Se and CryptoServer systems.

### 1.1.1 Target Audience

This document is primarily intended for all persons assuming the Global Administrator role for a u.trust Anchor Se device.

### 1.1.2 Document Conventions

We use the following document conventions:

<b>Convention</b>	<b>Use</b>	<b>Example</b>
<b>Bold</b>	Items of the Graphical User Interface (GUI), e.g., menu options	Press <b>OK</b>
<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.

### 1.1.3 Abbreviations

<b>Abbreviation</b>	<b>Description</b>
AES	Advanced Encryption Standard
BSI	Bundesamt für Sicherheit in der Informationstechnik (Federal Office for Information Security)
CA	Certificate Authority
CAK	Container Authentication Key
cHSM	Containerized Hardware Security Module
CNG	Cryptography API: Next Generation
CSP	Cryptographic Service Provider
CSR	Certificate Signing Request
CXI	Cryptographic eXtended Interface
csadm	CryptoServer command-line administration tool
DAK	Device Authentication Key
DES	Data Encryption Standard
DKMS	Dynamic Kernel Module Support
DMK	Device Master Key
DRBG	Deterministic random bit generator
DRNG	Deterministic random number generator
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve DSA
EKM	Extensible Key Management
FIPS	Federal Information Processing Standard
GAK	Global Admin Key

<b>Abbreviation</b>	<b>Description</b>
GAAC	Global Admin Authentication Key
GIAC	Global Initial Admin Key
gladm	Global Admin Management Tool
JCE	Java Cryptography Extension
JRE	Java Runtime Environment
MAC	Message authentication code
MBK	Master Backup Key
NTP	Network Time Protocol
P11CAT	PKCS#11 CryptoServer Administration Tool
PCIe	PCI Express Interface
PEM	Privacy-Enhanced Mail
PMU	Platform Management Unit
PRNG	Pseudorandom number generator
RSA	Rivest, Shamir, Adleman (cryptosystem)
SDMK	Sticky Device Master Key
TRNG	True random number generator

Table 2: Abbreviations

## 2 About the u.trust Anchor Se

### 2.1 Overview

The u.trust Anchor Se Containerized Hardware Security Module (cHSM) provides superior performance and GP HSM functionality. To reach this level of functionality, a cluster of cHSMs is running within u.trust Anchor Se, performing as one powerful cHSM.

When the u.trust Anchor Se device is set up, a cHSM is created in slot 1 of the device, and then duplicated to the remaining slots. All available commands and functions altering the data contained within the cHSM (e.g. for user management and key management) are automatically executed for all cHSMs, ensuring that the cluster operates as one powerful cHSM entity. There is no need to differentiate between the cHSMs for any of the functionalities supplied by u.trust Anchor Se cHSM device.

The number of cHSMs within the cluster depends on the u.trust Anchor Se model.

<b>Model</b>	<b>cHSMs</b>	<b>Performance</b>
u.trust Anchor Se100	1	RSA 100 sig/s, ECDSA 1000 sig/s
u.trust Anchor Se2k	4	RSA 2000 sig/s, ECDSA 4000 sig/s
u.trust Anchor Se5k	8	RSA 5000 sig/s, ECDSA 10000 sig/s
u.trust Anchor Se15k	4	RSA 15,000 sig/s, ECDSA 15,000 sig/s
u.trust Anchor Se40k	12	RSA 40,000 sig/s, ECDSA 40000 sig/s

Table 3: u.trust Anchor Se models



To upgrade your u.trust Anchor Se model, your u.trust Anchor device needs to be sent back to Utimaco to perform the upgrade, see [Contact Address for Support Queries](#).

#### 2.1.1 Hardware

Utimaco's hardware security module (HSM) u.trust Anchor is a physically protected specialized computer unit designed to perform sensitive cryptographic tasks and to securely manage cryptographic keys.

The diagram below shows the hardware components of u.trust Anchor located on the printed circuit board and completely covered by potting material. This hard, opaque enclosure protects the sensitive u.trust Anchor hardware components from physical attacks.



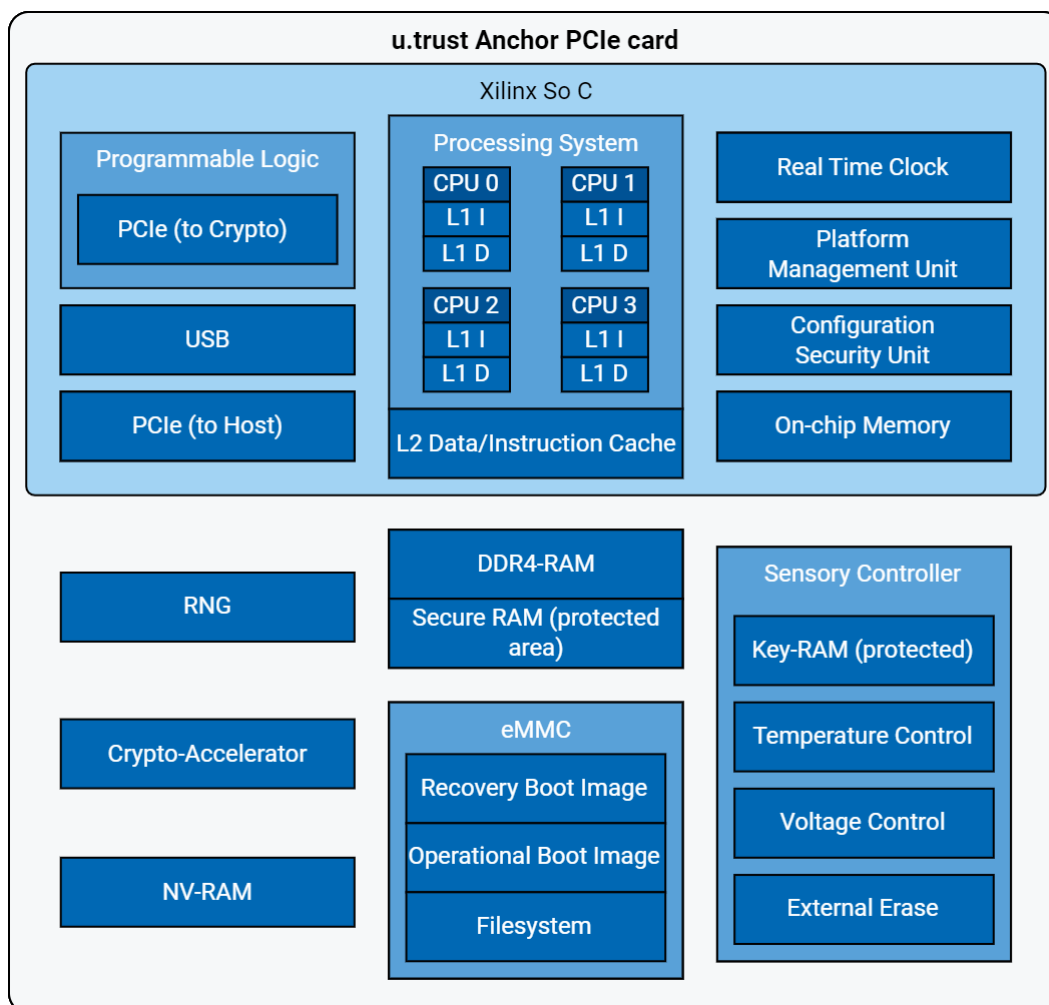


Figure 1 : u.trust Anchor - Hardware

### 2.1.1.1 Temperature-dependent Behavior of the u.trust Anchor Se Device

The u.trust Anchor device is fully operational only if its internal temperature does not exceed or fall below a well-defined operational temperature range. The next table shows how the u.trust Anchor's behavior changes depending on its internal temperature.



The operating temperature for u.trust Anchor ranges between -10 °C to 60 °C (50 °F to 140 °F).

<b><i>Internal Temperature</i></b>	<b><i>Behavior of u.trust Anchor</i></b>
Below -17 °C (1.4 °F)	Alarm is triggered, all sensitive data in the u.trust Anchor is deleted, and the device is restarted.
-17 °C to 77 °C (1.4 °F to 170.6 °F)	Normal operation
Above 77 °C (170.6 °F)	Alarm is triggered, all sensitive data in the u.trust Anchor is deleted, and the device is restarted.

Table 4: Operational temperature of u.trust Anchor

All temperature values in the table are approximate values. The exact temperature values may vary a little because of tolerances of the electronic components and the use of a hysteresis by the comparators.



Resetting the u.trust Anchor has no effect if its internal temperature still exceeds or is below the operational temperature range. In case of u.trust Anchor's constant high temperature, we recommend switching off the power supply for some time to cool down the u.trust Anchor.

Note that only the internal temperature of the u.trust Anchor Se is relevant, not the environmental temperature.

## 2.1.2 Software

During the boot process and the life cycle of a u.trust Anchor Se, several parts of its software come into action. This chapter only lists these different software components and gives a rough description of their functionality.

Inside the u.trust Anchor, different kinds of software run at different times:

- Bootloader,
- Operating system (SMOS) with firmware modules.

The bootloader is an independent firmware part that runs only partially on the u.trust Anchor whereas SMOS and the other modules run on the u.trust Anchor during its normal Operational Mode.

### 2.1.2.1 Boot Process

The bootloader is a special firmware that loads software images from persistent storage into RAM, verifies their correctness, and executes them. The bootloader is the first software started inside the u.trust Anchor after a power-up, reboot, or reset.

If during the boot process the bootloader finds itself initialized (i.e., the u.trust Anchor's public Production Key has been found) and the operating system module COSMOS is present in the u.trust Anchor, the latter will be started by the bootloader.



In case of an error during any of the boot stages, FSBL increments the multi-boot register, prints a message to the console, and resets the board (booting recovery).

A recovery boot is performed in the following cases:

- an alarm has been triggered,
- all operational images on the device are broken,
- after the External Erase button has been pressed for more than 10 seconds,
- or the reboot process has been interrupted several times in short order (a full reboot takes up to 50 seconds).



See also Leaving the Recovery Mode.

### 2.1.2.2 Operating System - SMOS Façade

The u.trust Anchor cHSM's operating system is called SMOS Façade (Security Module Operating System). It does not have direct access to the u.trust Anchor cHSM hardware and therefore directs all internal commands that require hardware access to COSMOS, the operating system running on the u.trust Anchor cHSM. It provides mechanisms for task handling, inter-process communication, memory management and file handling. Furthermore, SMOS Façade offers appropriate access command interfaces to the other firmware modules.

### 2.1.2.3 Firmware Modules

Firmware modules are an encapsulated software part running inside the u.trust Anchor cHSM. A module can have an external interface which can be used by an application from outside the

u.trust Anchor cHSM, and an internal C interface which can be called by other firmware modules.

The firmware modules can be divided into several classes:

- operating system (SMOS Façade),
- standard firmware modules provided by Utimaco
- other application firmware modules

The operating system and standard firmware modules are needed in order to get a running u.trust Anchor cHSM with basic functionality. Other application firmware modules are cryptographic interfaces like PKCS#11 or CXI and are provided by Utimaco.

The functional design of each standard firmware module and its interface is specified in more detail in the respective module-specific documentation, which is available with the SDK (Software Developer Kit) version of u.trust Anchor cHSMs.

With the exception of the command scheduler module CMDs, the administration module ADM, the cryptographic interface module CXI and the Master Backup Key management module MBK, these modules have no external interface and can only be accessed by other firmware modules via an internal public C-Interface.

A u.trust Anchor cHSM generally stores a firmware module in the form of a MSC (module storage container). The MSC format is for u.trust Anchor cHSM-internal use only. It stores the module code together with certain module information and the check value for the module code's integrity (SHA-512 hash value over the module code which will be verified at every start-up).

### **Failing of Firmware Modules**

If the module CMDs or any of its dependencies fail to start, the u.trust Anchor cHSM will enter dead state and not respond to any commands.

If the module CRYPT fails to start, a cHSM will not respond to any commands, while a FIPS cHSM will enter the ERROR state.

If other modules fail to start, the u.trust Anchor cHSM responds to commands but won't register the external interface of the module, i.e. if the module CXI fails to start, the u.trust Anchor cHSM will perform CMDs commands, but CXI commands will not be available and an error message will be returned instead.

### 2.1.2.3.1 Firmware Module Management

It is only possible to load firmware modules provided and signed by Utimaco. It is not possible to load firmware modules from any other source.

With the aim to link the firmware with administrative information (for example, module name and version number) and to add check values for the authenticity and integrity of the firmware, every firmware module is enveloped into a so-called container. These containers allow also the load and storage of encrypted firmware.

Utimaco can supply the u.trust Anchor user with firmware modules signed by Utimaco. They will be provided in `.mtc` format.

- MTC

An MTC (Module Transport Container) is used to secure the transport of the RFM from the developer to the customer and in this way guarantee the authenticity of the RFM during delivery. The MTC adds an MTC header, which contains additional module transport information, and an MTC signature for secure and authentic transport. The MTC header is mandatory but the MTC signature is optional. An MTC is stored as an `*.mtc` file.

- MSC

After the MTC has been transported to and loaded into the u.trust Anchor cHSM, the u.trust Anchor cHSM unwraps the RFM from the MTC and stores it as an MSC (Module Storage Container).

The MSC format is for u.trust Anchor cHSM-internal use only. It stores the RFM together with certain module information and the check value for the RFM's integrity (SHA-512 hash value calculated over the RFM). All sections of an MSC except the type section are mandatory.

Perform the `csadm ListFiles` command to show all available `*.msc` files on a u.trust Anchor cHSM.

To show the contents of a firmware module, perform the `csadm ModuleInfo` command. See *Module Info* in the [u.trust Anchor - csadm Manual](#), for details.

## 2.1.3 Administration Tools

The u.trust Anchor device can be administered with three command-line tools, the CryptoServer administration tool *csadm*, the Global Administration Management (*gladm*) tool, and the PKCS#11 Administration Tool *p11toolv2*.

The functionality of the administration tools is explained briefly in the following sections. For detailed command references, please refer to the respective manuals, [u.trust Anchor - csadm Manual](#), the [u.trust Anchor - Administration Manual](#), and [CryptoServer – PKCS#11 p11tool2 Reference Manual](#).

### 2.1.3.1 CryptoServer Command-Line Administration Tool (csadm)

The u.trust Anchor Se device is administered with the CryptoServer command line tool *csadm*.



This manual describes the general context of *csadm* and the functionality of commands included in specific procedures to administer the device. For a complete overview of all *csadm* installation, syntax, commands, and their parameters, refer to the [u.trust Anchor - csadm Manual](#).

The CryptoServer command-line tool *csadm* is a program designed for the administration of the device that can be called from either a command line or a batch file. It handles all typical administration tasks like 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.

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).

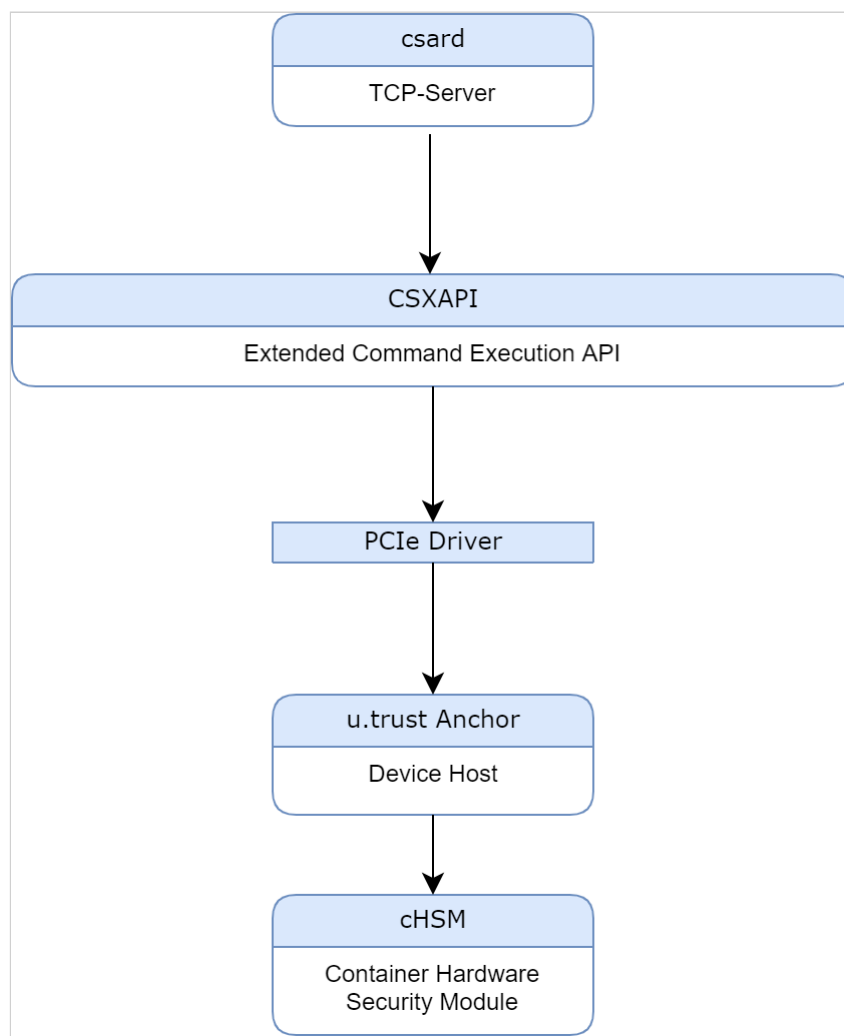


Figure 2 : u.trust Anchor csadm

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:

<b>Command Destination</b>	<b>Counterpart on the cHSM</b>	<b>Command Group</b>
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 <i>csadm</i> )
TCP server of the device host	Control module of the TCP Server (daemon)	Administration (configuration) of the TCP Server ('daemon')

Table 5: Command classes

### 2.1.3.2 gladm Administration Tool

u.trust Anchor Se offers access to the administration tool *gladm*, see the [u.trust Anchor - Administration Manual](#). While most of the administrative tasks on the u.trust Anchor Se device have to be conducted with *csadm*, it is necessary to use the administration tool *gladm* for the following:

- requesting the alarm state, *Displaying Device System Information (system-info)*,
- clearing the device, *Clearing the System (system-clear)*,
- resetting the alarm, *Resetting the Alarm (system-reset-alarm)*.

This manual provides a detailed description of all available *gladm* commands with the following information:

- **Command**  
The command in correct syntax with all available arguments;
- **Arguments**  
All arguments available for the command listed in their available syntax, including a detailed description, possible values and dependencies, optional arguments are given in square brackets;



- **Example**

An example of the command in a use case is provided where possible;

- **Return**

The return of the command upon successful execution. If a command does not have any return parameters, nothing will be returned upon successful execution.

### Inline Help

In addition to these descriptions, an inline help can be called at any time, providing a list of all available commands. It can be accessed via the following command:

```
gladm -h
```

It can also be used following a specific command, providing a description of the command along with a list and short description of the available arguments:

```
gladm chsm-create -h
```

#### 2.1.3.2.1 Specifying a Device

The u.trust Anchor device to which a command is to be sent is specified by the global `-d` argument (device specifier/address). This argument is mandatory for all gladm commands except for the `gladm bash-completion` command where the device specifier/address is unnecessary because, in this case, a connection to the u.trust Anchor PCIe card is not needed. The device specifier must be used regardless of whether the u.trust Anchor device is accessed locally or remotely or whether the u.trust Anchor PCIe card is mounted in a Linux or a Windows computer or in a u.trust Anchor LAN device.

There is no default value for the device specifier. If no value is specified, the following error message is output:

```
gladm error: No device was given. Please specify the device using the device specifier.
```

The device specifier may have the following values:

- Local access

- Local access via the device node

In this case, no port is used because the communication is not transmitted via the network. The `-p <port>` argument is not needed. No default value of this argument is used, i.e., the `Port` parameter in the `[ListenerGlad]` section of the `csard.conf` configuration file or the `/etc/csxlان.conf` configuration file is not used. If the `-p` argument is set in the `gladm` command (example: `gladm ... -d /dev/cs2.0 -p 5555 ...`), it is ignored.

- Specifying a u.trust Anchor PCIe card mounted in a local Linux computer or a local u.trust Anchor LAN device:

```
gladm ... -d /dev/cs2.0 ...
```

- Specifying a u.trust Anchor PCIe card mounted in a local Windows computer:

```
gladm ... -d PCI:0 ...
```

- Local access via the network

Specifying a u.trust Anchor PCIe card mounted in a local Linux/Windows computer or a local u.trust Anchor LAN device

Examples:

- `gladm ... -d 127.0.0.1 ...`
- `gladm ... -d 127.0.0.1 -p 4000 ...`
- `gladm ... -d 127.0.0.1 -p 5555 ...`
- `gladm ... -d localhost ...`
- `gladm ... -d localhost -p 4000 ...`
- `gladm ... -d localhost -p 5555 ...`

- Remote access

Specifying a u.trust Anchor PCIe card mounted in a remote Linux/Windows computer or a remote u.trust Anchor LAN device

- Using the IP address of the remote computer or LAN device:

Examples:

- `gladm ... -d 123.123.123.123 ...`
- `gladm ... -d 123.123.123.123 -p 4000 ...`

- `gladm ... -d 123.123.123.123 -p 5555 ...`
- Using the hostname of the remote computer or LAN device  
Examples:

- `gladm ... -d myPC ...`
- `gladm ... -d myPC -p 4000 ...`
- `gladm ... -d myPC -p 5555 ...`
- `gladm ... -d myUtaLan ...`
- `gladm ... -d myUtaLan -p 4000 ...`
- `gladm ... -d myUtaLan -p 5555 ...`

The optional `-p` argument indicates the port that is used on the computer or LAN device. Default if this argument is not set: 4000. If you use a u.trust Anchor PCIe card mounted in a Linux/Windows computer, this default value is configurable by the `Port` parameter in the `[ListenerGlad]` section of the `csard.conf` configuration file of the CSAR daemon (csard). If you use a u.trust Anchor LAN device, this default value is configurable by the `Port` parameter in the `[ListenerGlad]` section of the `/etc/csxlan.conf` configuration file of the u.trust Anchor LAN device.

Example of the configuration file:

```
...
[CryptoServerGlad]
Label = GladCS1
Timeout = 30000
Device = /dev/cs2.0

[ListenerGlad]
Port = 4000
Keepalive = 1
Route_To = GladCS1
...
```

Do not confuse this `-p` argument indicating a port with the following `-p` arguments in the following gladm commands:

- `gladm ... user-add [-p <val>] ...`
- `gladm ... chsm-create ... [-p <directory_path>] ...`
- `gladm ... system-get-trust-chain [-p <directory path>]`

The `-p` argument indicating a port is identified by gladm by the position of the `-p` argument within the gladm command. For example, a command

`gladm -d <device specifier/address> -p <port> ... user-add [-p <val>] ...`  
is supported, but a command

```
gladm ... user-add [-p <val>] ... -d <device specifier/address> -p <port>
```

is not.

### 2.1.3.3 p11tool2 Administration Tool

The *PKCS#11 Administration Tool p11toolv2* (p11tool2) is a command-line tool designed for being called from the command line or in a batch file. It offers various functions to execute PKCS#11 typical key management and configuration commands on the cHSM. The p11toolv2 is mainly created to support the cHSM's Security Officers and Cryptographic Users or Key Managers when performing PKCS#11 typical tasks:

- An operator who is allowed to assume the cHSM Security Officer role can use p11toolv2 to set up and display group (slot) specific configuration values and to generate or delete the PKCS#11 standard slot User.
- An operator who is allowed to assume the cHSM Administrator role can use p11toolv2 to set up and display global (not group or slot specific) configuration values and to generate or delete the PKCS#11 standard slot Security Officers.
- In the default configuration of the cHSM (configuration attribute `CKA_CFG_AUTH_KEYM_MASK` set to 00000002) every user with permission mask 00000002' (or higher) is allowed to assume the Cryptographic User role (User and Key Manager role) and can use the PKCS#11 administration tool p11toolv2 to execute key management services like key generation or key backup and restore, and to display the current configuration setting.
- If the configuration attribute `CKA_CFG_AUTH_KEYM_MASK` is set to 00000020, key management services can only be executed by dedicated Key Managers (Users with a permission mask of '00000020' or higher). In this case, with p11toolv2 the PKCS#11 standard slot User can only list available keys and storage objects and display the current configuration settings.

The [u.trust Anchor - PKCS#11 p11tool2 - Reference Manual](#) gives a detailed description of installation and usage of p11toolv2.

If one of the users in the user roles uses the RSA Signature authentication mechanism for authenticating security-relevant PKCS#11 commands, and the private part of his RSA authentication key is stored on a smartcard, the PIN pad must be connected to the administration computer on which the p11tool2 has been installed.

#### Remarks

- A *slot* according to PKCS#11 corresponds to Key Group `SLOT_<nnnn>`.
- The Security Officer (SO) for slot `<nnnn>` according to PKCS#11 corresponds to user `SO_<nnnn>` with permissions 00000200 (for example, Security Officer `SO_0001` for slot 1 resp. Key Group `SLOT_0001`).
- The slot user according to PKCS#11 for slot `<nnnn>` corresponds to user `USER_<nnnn>` with permission mask '00000002' (for example, user `USER_0001` for slot 1 resp. Key Group `SLOT_0001`).
- Users with different user names, Key Groups or permission masks (for example, Key Managers with permissions 00000020) must be configured by a CHSM Administrator using the appropriate `csadm` commands.

## 3 Backup and Restore

### 3.1 Preconditions



For a smooth migration from CryptoServer to u.trust Anchor, the same firmware version must be installed on both systems.

Use the command `csadm ListFirmware` to check the firmware version.

This function returns a list with information about all firmware modules that are currently running, giving the firmware module ID, abbreviated name, CPU target `type` the firmware module is compiled for ( `C64+` for CSe-Series, `C64` for Se-Series Gen2, `SDK` for Simulator for Windows and `SIM` for Simulator for Linux), version number and the respective firmware module `initialization level`.



If a module cannot be started or fully initialized, the `csadm GetBootLog` command provides more detailed information about the reason.

<b>Syntax</b>	<code>csadm [Dev=&lt;device&gt;] ListFirmware</code>
<b>Parameter</b>	<b>Description</b>
<device>	Device address This parameter can be omitted if the device address has been set as the environment variable CRYPTOSERVER.
<b>Example</b>	<code>csadm Dev=192.168.1.1 ListFirmware</code>

Upon successful execution of the command, a list of all firmware modules is returned. The example output may differ from the output on your CryptoServer with respect to the listed firmware modules and/or their versions depending on the installed firmware package and its version.

If for a firmware module no entry is found, the module is not loaded.



After loading or replacing a firmware module, the device has to be restarted with the `Restart` command before the firmware module becomes active.

For CryptoServer 4.45 and later, the following applies:

The implementation of cryptographic primitives has been optimized for, and harmonized across, Utimaco's HSM platforms and host components, resulting in a general performance increase of cryptographic operations. The exact improvements depend on algorithm, key size, size of data to be encrypted/signed, and HSM model. The improvements are highest for short-running operations like AES encryption of small blocks of data. This new common cryptographic library is delivered as CRYPT firmware module. In a firmware package, this CRYPT firmware module replaces the following modules that were previously handled as separate modules: AES, HASH, DSA, VRSA, LNA, ECDSA, ECA, POST and ASN.1. The modules AES, HASH, etc. appear as active firmware modules in addition to the new CRYPT module when calling the `csadm ListFirmware` command.

Example output for CryptoServer 4.45 and later:

ID	name	type	version	initialization	level
0	SMOS	SDK	5.6.4.1	INIT_OK	
4	POST	SDK	2.2.1.0	INIT_OK	
68	CXI	SDK	2.4.11.1	INIT_OK	
81	VDES	SDK	2.2.1.0	INIT_OK	
82	PP	SDK	1.4.1.2	INIT_OK	
83	CMDS	SDK	3.8.4.1	INIT_OK	
84	VRSA	SDK	2.2.1.0	INIT_OK	
85	SC	SDK	1.2.0.7	INIT_OK	
86	UTIL	SDK	3.0.7.1	INIT_OK	
87	ADM	SDK	3.1.2.0	INIT_OK	
88	DB	SDK	2.0.0.2	INIT_OK	
89	HASH	SDK	2.2.1.0	INIT_OK	
8a	STUN	SDK	0.0.0.1	INIT_OK	
8b	AES	SDK	2.2.1.0	INIT_OK	
8d	DSA	SDK	2.2.1.0	INIT_OK	
8e	LNA	SDK	2.2.1.0	INIT_OK	
8f	ECA	SDK	2.2.1.0	INIT_OK	
91	ASN1	SDK	2.2.1.0	INIT_OK	
96	MBK	SDK	2.5.1.1	INIT_OK	
9a	NTP	SDK	1.2.1.1	INIT_OK	
9c	ECDSA	SDK	2.2.1.0	INIT_OK	
9f	CRYPT	SDK	2.2.1.0	INIT_OK	

Example output for CryptoServer earlier than 4.45:

### Output

ID	name	type	version	initialization level
0	SMOS	C64	3.1.1.2	INIT_OK
a	HCE	C64	1.0.1.0	INIT_OK
e	BCM	C64	1.0.2.0	INIT_OK
64	PKCS11	C64	1.1.3.0	INIT_OK
68	CXI	C64	2.1.0.1	INIT_OK
81	VDES	C64	1.0.4.0	INIT_OK
82	PP	C64	1.2.3.6	INIT_OK
83	CMDS	C64	3.0.3.5	INIT_OK
84	VRSA	C64	1.1.1.2	INIT_OK
85	SC	C64	1.2.0.0	INIT_OK
86	UTIL	C64	3.0.1.2	INIT_OK
87	ADM	C64	3.0.7.1	INIT_OK
88	DB	C64	1.1.2.4	INIT_OK
89	HASH	C64	1.0.7.0	INIT_OK
8b	AES	C64	1.0.5.0	INIT_OK
8d	DSA	C64	1.0.0.0	INIT_OK
8e	LNA	C64	1.1.0.0	INIT_OK
8f	ECA	C64	1.1.2.0	INIT_OK
91	ASN1	C64	1.0.3.3	INIT_OK
96	MBK	C64	2.2.3.2	INIT_OK
9a	NTP	C64	1.2.0.6	INIT_OK
9c	ECDSA	C64	1.1.1.1	INIT_OK

The meaning of the initialization levels is the following:

<b>Initialization level</b>	<b>Description</b>
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 SD-RAM memory (for CSe-Series: DDR2 RAM memory).
INIT_INTERNAL	The module has finished its internal initialization (first step of 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 depends on services provided by other modules in order to run correctly. Example: the SC (Smartcard) module requires the PP (PIN pad) module to do its work.
INIT_OK	This is the highest possible level: The initialization of the module is successfully 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.
INIT_INACTIVE	Module is loaded but cannot supply services, for example, because of not-mounted hardware components. This initialization level is currently only used by the firmware modules HCE and EXAR (relevant for Se-Series Gen2 only).
SUSPENDED	Module was shut down and cannot supply services anymore.

Table 6: Initialization levels



## 3.2 Backing up Databases on CryptoServer SeGen2

### 3.2.1 Generate Master Backup Key (MBK)

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.

<b>Syntax</b>	<code>csadm [Dev=&lt;device&gt;] &lt;Authentication&gt;Key=&lt;keyspec&gt; MBKGenerateKey=&lt;keytype&gt;,&lt;keylen&gt;[,&lt;n&gt;,&lt;m&gt;,&lt;keyname&gt;]</code>
---------------	---

<b>Authentication</b>	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"> <li>Key specifier where the generated MBK shares should be stored <code>&lt;smartcard,record number&gt;</code>, e.g. <code>:cs2:cjo:USB0,15</code> The default record number for AES keys is 15.</li> <li>List of filenames and (if the key should be password protected) passwords, <code>&lt;file#password&gt;</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.</li> </ul>
<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
<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**

- `csadm LogonSign=ADMIN,:cs2:cjo:USB0 Key=:cs2:cjo:USB0,1  
MBKGenerateKey=AES,24,2,2,MyAESKey`
- `csadm LogonPass=Paul,mypassword  
Key=mbk1.key#swordfish,mbk2.key#sesame  
MBKGenerateKey=AES,32,2,2,MyAESKey`
- `csadm LogonPass=Paul,ask Key=:cs2:cjo:USB0,15  
MBKGenerateKey=AES,32,5,3,AES_new`

**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.

## 3.2.2 Backup User Database

### 3.2.2.1 Backup User with csadm 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.

### 3.2.2.2 Backup with csadm BackupUser

This command creates a file with a backup of all users that currently exist on the device. Each user entry in the created backup file is encrypted with the device 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.

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 *ListUser*.


#### Syntax

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

#### Authentication

The command must be authenticated with permission 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

Parameter	Description
<flags>	<p>Overwrite (Optional) Backup file is overwritten if already existing</p> <hr/> <p> If the 'overwrite' flag is not set and the backup file already exists, no backup is performed and the existing backup file is not overwritten.</p> <hr/>

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

<b>Output</b>	Upon successful execution of the command, no output is given.
---------------	---

### 3.2.3 Backup Key Database

#### 3.2.3.1 Backup Key with csadm 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>
```

<b>Authentication</b>	Permission level 2 in user group 6 and 7 (22000000)
-----------------------	---

<b>Parameter</b>	<b>Description</b>
<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

<b>Example</b>	Back up the cryptographic key database <code>csadm LogonSign=ADMIN,:cs2:cjo:USB0 BackupDatabase=CXIKEY.db</code>
----------------	---

<b>Output</b>	Upon successful execution of the command, no output is given.
---------------	---

### 3.2.3.2 Backup Key with cxitool Backupkey

This command backs up one or several cryptographic keys from the CryptoServer's internal key database, `CXIKEY.db`, or an external keystore to a keyfile ( `*.kbk` ). An already existing keyfile is overwritten. This keyfile is encrypted with the CryptoServer's Master Backup Key (MBK; default) or a Tenant Backup Key (TBK). A TBK is enabled by performing the `cxitool SetConfig` command and setting the `SecureGroupBackup` parameter to true.



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

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 or after an MBK rollover in an MBK slot  $\geq 3$ . Note down this MBK as well if a TBK is used.

Otherwise, the backup file is inaccessible. This might be the result of the execution of a `csadm MBKImportKey` command, see *Master Backup Key Rollover* in the *CryptoServer – Administration Manual*.

It is not possible to retrieve the MBK by which a backup file has been generated from this backup file.





If the `Export` parameter of a key has been set to `deny_backup` value (`0x00010000`), the `cxitool BackupKey` command cannot be performed. However, the `csadm BackupDatabase` command can be performed. The `deny_backup` value cannot be set by the `cxitool Export` parameter. It can be set only by the CXI API. However, it can be shown by the `cxitool KeyInfo` command. For details, see section *KeyInfo* in *CryptoServer - cxitool Manual*.

Export properties can only be set when generating/importing/copying an object. They cannot be changed afterwards.

### Syntax

```
cxitool [Dev=<dev>] <auth> [Name=<name>] [Group=<group>]
[Spec=<spec>] [KeyStoreType=<keystoretype>]
KeyStoreParam=<keystoreparam> [OutDir=<outdir>] BackupKey
```

### Authentication

- User manager, level 2 in user group 7, 20000000: Backing up the global configuration
- Security officer (SO, key group manager), level 2 in user group 2, 00000200: Backing up the key group configuration
- Key manager, level 2 in user group 1, 00000020, or level 2 in user group 0, 00000002 (default): Backing up a cryptographic key

The following applies to the security officer and the key manager:  
Ensure that the authenticated user is assigned ( `CXI_GROUP` user attribute) to the key group(s) specified by the `<group>` parameter. For more information, see section *Cryptographic Keys and Key Groups* in the *CryptoServer - cxitool Manual*.  
A user is assigned to a key group on creation. Use the `csadm ListUser` command or open **Manage > User** in the CryptoServer Administration Tool (CAT) to verify the key group the user is assigned to. For details, see the *CryptoServer - csadm Manual* and the *CryptoServer - CAT Manual*.  
Key groups must not be confused with user groups.

### Parameter

### Description

`<dev>`

Device address of the CryptoServer or the CryptoServer LAN. This parameter can be omitted if the device address has been set as the CRYPTOSERVER environment variable.

Parameter	Description
<name>	<p>Cryptographic key name At least one of &lt;name&gt;, &lt;group&gt; or &lt;spec&gt; is mandatory. The * and ? wildcards are supported. If Name="" or Name=, only keys without a name are backed up. Combinations of empty names and groups with wildcards are supported, for example:</p> <ul style="list-style-type: none"> <li>Group="1234" and Name="*" backs up all keys of group 1234.</li> <li>Group="1234" and Name="" backs up all keys with empty names of group 1234.</li> </ul> <p>If you want to back up a global configuration object or a key group configuration object, omit the Name parameter because configuration objects do not have names. As described above, make sure to use a user with the correct permission mask (20000000 for backing up the global configuration object or 00000200 for backing up the key group configuration object) for the authentication of the command.</p>
<group>	<p>Key group name for cryptographic key. If Group="" or Group=, only a global configuration object (authenticated user with the permission mask 20000000) or keys without a group are backed up. Again, combinations of empty names and groups and with wildcards are supported, for example:</p> <ul style="list-style-type: none"> <li>Group="1234" and Name="*" backs up all keys of group 1234.</li> <li>Group="1234" and Name="" backs up all keys with empty names of group 1234.</li> </ul>
<spec>	Cryptographic key specifier
<keystoretype>	<p>Type of an external keystore Its value is either SDB (for legacy .sdb files) or ODBC. If KeyStoreType is used, KeyStoreParam must be used as well. If KeyStoreType is used, it must be used before KeyStoreParam. If KeyStoreType is used, this cxitool command only lists the keys in the external keystore. In this case, an unspecified key specifier is shown as empty. If KeyStoreType is not used, this cxitool command only lists keys the CryptoServer's internal key database, CXIKEY.db.</p>
<keystoreparam>	<p>Configuration parameter for an external keystore If KeyStoreType is used, KeyStoreParam must be used as well. If KeyStoreType is used, it must be used before KeyStoreParam.</p> <ul style="list-style-type: none"> <li>If KeyStoreType=SDB KeyStoreParam specifies the path to an external keystore file ( /path/to/store.sdb ).</li> <li>If KeyStoreType=ODBC KeyStoreParam specifies the data source name as defined in format DSN=DATA SOURCE NAME or the complete ODBC connection string.</li> </ul>

Parameter	Description
<outdir>	<p>Name of the directory the keyfile ( *.kbk file) is stored in. The current directory of the command line is the default directory.</p> <p>The name of a keyfile is built according to the following pattern:  [&lt;group&gt;_] [&lt;name&gt;_] [&lt;spec&gt;] .kbk  The key group name is omitted if it is not available. The key name is omitted if it is not available.</p>
Example	<p>Example 1: Backing up one cryptographic key specified by Group=1234 Name=RSA2048</p> <pre>cxitool Dev=3001@127.0.0.1 LogonPass=KeyMgr1234,12345678 Group=1234 Name=RSA2048 OutDir=C:\Utimaco\CryptoServer BackupKey</pre> <p>Example 2: Backing up all cryptographic keys of Group SLOT_0000</p> <pre>cxitool Dev=3001@127.0.0.1 LogonPass=KeyMgrSlot0,12345678 Group=SLOT_0000 OutDir=C:\Utimaco\CryptoServer BackupKey</pre>
Output	<p>Upon successful execution of the command, the location of the backup file along with the backed up keys or key groups are returned.</p> <p>Example 1:</p> <pre>C:\Utimaco\CryptoServer\1234_RSA2048.kbk 1 key(s) backed up</pre> <p>Example 2:</p> <pre>C:\Utimaco\CryptoServer\SLOT_0000_9.kbk C:\Utimaco\CryptoServer\SLOT_0000_1.kbk C:\Utimaco\CryptoServer\SLOT_0000_9.kbk 3 key(s) backed up</pre>

### 3.2.3.3 Backup Key with p11tool2 BackupInternalKeys

This command backs up all available internal keys within a PKCS#11 slot.



Perform the `csadm MBKListKeys` command to determine which Master Backup Key (MBK) is currently in use in MBK slot 3 by the CryptoServer. This MBK is used by the `p11tool2 BackupInternalKeys` command to encrypt the backup file to be generated. If the MBK in MBK slot 3 is the autogenerated MBK named `AUTO-GEN`, the `p11tool2 BackupInternalKeys` command cannot be performed. Import a different MBK into MBK slot 3 using the `csadm MBKImportKey` command described in [CryptoServer - csadm Manual](#).

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 or after an MBK rollover in an MBK slot  $\geq 3$ .

Otherwise, the backup file is inaccessible. This might be the result of the execution of a `csadm MBKImportKey` command. See Chapter "Master Backup Key Rollover" in [CryptoServer - csadm Manual](#) for details.

It is not possible to retrieve the MBK by which a backup file has been generated from this backup file.

<b>Syntax</b>	<code>p11tool2 [Slot=&lt;slot_id&gt;] [Force=&lt;force&gt;] &lt;login_key_manager&gt; BackupInternalKeys=&lt;filename&gt;</code>
---------------	--

Parameter	Description
<slot_id>	ID of the PKCS#11 slot as number Default: 0
<force>	Boolean flag (0/1 or n/y) to overwrite file if already exists. Default: 0 (Cancel command if file already exists)
<login_key_manager> >	Login as key manager (via Login). Note: By default the key manager and the key user have the same permission mask. In that case the normal user can also be logged in (via LoginUser).
<filename>	Name of the key backup file to be created

<b>Example</b>	<code>p11tool2 Slot=1 Login=keyM,ask BackupInternalKeys=C:/backup/internal_keys_p11slot1.bak</code>
----------------	---

<b>Output</b>	<code>16 internal key(s) backed up</code>
---------------	---

## 3.3 Restoring Databases on u.trust Anchor Se

### 3.3.1 Import Master Backup Key (MBK)

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.

<b>Syntax</b>	<code>csadm [Dev=&lt;device&gt;] &lt;Authentication&gt; Key=&lt;keyspec&gt; MBKImportKey=&lt;slot_no&gt;</code>
<b>Authentication</b>	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 <b>&lt;file#password&gt;</b> 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.
<b>Example</b>	<code>csadm LogonPass=Paul,&lt;Paul key file&gt; ask Key=mbk1.key#swordfish,mbk2.key#sesame MBKImportKey=3</code>

**Output**

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

### 3.3.2 Restore User Database

#### 3.3.2.1 Restore User with csadm 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.



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 CRYPTOSEVER.
<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.

### 3.3.2.2 Restore User with csadm 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 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.

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 user3 (B0890007)  
Skipped user user7 (B0890007)  
Skipped user Admin (User is currently logged in)
```

#### Syntax

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



#### Authentication

The command must be authenticated with permission 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

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 ‘&lt;user name&gt;’ [error code]</code> and <code>FC:0x083 SFC:0x0D Restore User ‘&lt;user name&gt;’ (&lt;short user name&gt;, &lt;authentication mechanism ID&gt; &lt;user’s permission&gt; &lt;access properties&gt;) [error code]</code> <code>&lt;access properties&gt;</code> contains a comma-separated list of strings of the form <code>&lt;name&gt;=&lt;value&gt;</code> . Examples: <code>CXI_GROUP=SLOT_0001</code> <code>CXI_GROUP=test</code> <b>add (default)</b>															
	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>&lt;username&gt;</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>&lt;username&gt;</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 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>&lt;username&gt;</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>&lt;username&gt;</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 backup file.	None	None	None												
	<b>overwrite</b>															
	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>&lt;username&gt;</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> and <code>Restore User</code>	User <code>&lt;user name&gt;</code> replaced in <code>user.db</code>				
	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>&lt;username&gt;</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> and <code>Restore User</code>	User <code>&lt;user name&gt;</code> replaced in <code>user.db</code>													

Parameter	Description			
	Scenario	Action	Audit Log Entry	Output
	User in the backup file already exists with in the <code>user.db</code> database and user is logged in.	None	None	User <user name> not replaced in <code>user.db</code> because the user is currently logged in
	User exists in the <code>user.db</code> database but not in the backup file.	None	None	None
	replace			
	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.	Restore User	User <user name> added to <code>user.db</code>
	User in the backup file already exists in the <code>user.db</code> database.	User is overwritten.	Delete User and Restore User	User <user name> replaced in <code>user.db</code>
	A 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.	<code>error number != 0</code> and The restore operation is aborted and not a single user account is restored.	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 backup file.	User is deleted. If this user is a single logged-in user manager, deleting this user manager is the last operation of the procedure, because deleting this user manager terminates the Secure Messaging session.	Delete User	User <user name> removed from HSM

Parameter	Description
	<div>  <p>If the deleted user is 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.</p> </div> <div>  <p>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.</p> </div>

<b>Example</b>	<code>csadm LogonSign=ADMIN,:cs2:cjo:USB0 RestoreUser=d:\temp\cs123456-20051212.ubk,overwrite</code>
<b>Output</b>	Upon successful execution of the command, an indication of the executed action is returned as stated in the flag table above.

### 3.3.3 Restore Key Database

#### 3.3.3.1 Restore Key with `csadm 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.



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>
```

<b>Authentication</b>	Permission level 2 in user group 6 and 7 (22000000)
-----------------------	---

<b>Parameter</b>	<b>Description</b>
<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)

<b>Example</b>	Restore the cryptographic key database csadm LogonSign=ADMIN,:cs2:cjo:USB0 RestoreDatabase=CXIKEY.db
----------------	--

<b>Output</b>	Upon successful execution of the command, no output is given.
---------------	---

### 3.3.3.2 Restore Key with cxitool RestoreKey

A cryptographic key may have been backed up to a keyfile ( \*.kbk ). This keyfile is encrypted with a Master Backup Key (MBK; default) or a Tenant Backup Key (TBK). A TBK is enabled by performing the `cxitool SetConfig` command and setting the `SecureGroupBackup` parameter to true.

If this MBK or TBK is available in the CryptoServer, the `cxitool RestoreKey` command can restore the cryptographic key from the keyfile to the internal key database of the CryptoServer, `CXIKEY.db`, or an external keystore. Thus, multiple CryptoServers can be aligned, for example, to contain the same CA's signature key.

All parameters but the filename of the backup file are optional if the CRYPTOSERVER variable has been set, otherwise the device parameter is mandatory. The optional parameters can be specified if other attributes than the ones saved in the backup file should be assigned to the restored key. If a cryptographic key with the same Group, Name and Spec parameters is already available in the internal key database or the external keystore, it is overwritten.

The `cxitool RestoreKey` command does not support the Usage parameter. See section *Usage* in [CryptoServer - cxitool Manual](#) for details. The usage information that is stored in the keyfile is restored.

<b>Syntax</b>	<code>cxitool [Dev=&lt;dev&gt;] &lt;auth&gt; [Name=&lt;name&gt;] [Group=&lt;group&gt;] [Spec=&lt;spec&gt;] [KeyStoreType=&lt;keystoretype&gt; KeyStoreParam=&lt;keystoreparam&gt;] RestoreKey=&lt;filename&gt;</code>
---------------	---

<b>Authentication</b>	<ul style="list-style-type: none"> <li>▪ User manager, level 2 in user group 7, 20000000: Backing up the global configuration</li> <li>▪ Security officer (SO, key group manager), level 2 in user group 2, 00000200: Backing up the key group configuration</li> <li>▪ Key manager, level 2 in user group 1, 00000020, or level 2 in user group 0, 00000002 (default): Backing up a cryptographic key</li> </ul> <p>The following applies to the security officer and the key manager: Ensure that the authenticated user is assigned ( <code>CXI_GROUP</code> user attribute) to the key group(s) specified by the <code>&lt;group&gt;</code> parameter. For more information, see section <i>Cryptographic Keys and Key Groups</i> in the <i>CryptoServer - cxitool Manual</i>. A user is assigned to a key group on creation. Use the <code>csadm ListUser</code> command or open <b>Manage &gt; User</b> in the CryptoServer Administration Tool (CAT) to verify the key group the user is assigned to. For details, see the <i>CryptoServer – csadm Manual</i> and the <i>CryptoServer – CAT Manual</i>. Key groups must not be confused with user groups.</p>
-----------------------	--

<b>Parameter</b>	<b>Description</b>
<dev>	Device address of the CryptoServer or the CryptoServer LAN. This parameter can be omitted if the device address has been set as the CRYPTOSERVER environment variable.
<name>	Cryptographic key name that shall be used to store the restored cryptographic key in the CryptoServer's internal database, <code>CXIKEY.db</code> , or in an external keystore. This name does not need to match the cryptographic key name that is part of the name of the kbk file containing the key to be restored. The * and ? wildcards are not supported. If <code>&lt;name&gt;</code> is omitted, the name of the backed up key is assigned to the restored key.
<group>	The key group the restored cryptographic key is assigned to If the key group of the backed up key differs from the Group parameter in the <code>cxitool RestoreKey</code> command, ensure that the user authenticating this command has been assigned to both key groups using wildcards. As an alternative, you can use two users for authenticating this command with one user being assigned to the key group of the backed up cryptographic key and the other user being assigned to the key group of the restored cryptographic key. If these conditions are not fulfilled, a permission denied error message is shown. If the <code>&lt;group&gt;</code> parameter is omitted in the <code>cxitool RestoreKey</code> command, the cryptographic key is automatically assigned to the group stored in the backup file. If the cryptographic key has been backed up without being assigned to any group, the cryptographic key can be restored without being assigned to any group or with being assigned to an arbitrary group. Consider in the latter case that the user performing the restore command must have been assigned to the corresponding key group. If the group name had been available for the key backup, the group name is part of the keyfile name. If <code>&lt;group&gt;</code> is omitted, the key group name of the backed up key is assigned to key group name of the restored key.

Parameter	Description
<spec>	Cryptographic key specifier that shall be used to store the restored cryptographic key in the CryptoServer's internal database, <code>CXIKEY.db</code> , or the external keystore. This cryptographic key specifier does not need to match the key specifier that is part of the name of the kbk file containing the key to be restored. If <spec> is omitted, the key specifier of the backed up key is assigned to key specifier of the restored key.
<keystoretype>	Type of an external keystore Its value is either SDB (for legacy .sdb files) or ODBC. If <code>KeyStoreType</code> is used, <code>KeyStoreParam</code> must be used as well. If <code>KeyStoreType</code> is used, it must be used before <code>KeyStoreParam</code> . If <code>KeyStoreType</code> is used, this cxitool command only lists the keys in the external keystore. In this case, an unspecified key specifier is shown as empty. If <code>KeyStoreType</code> is not used, this cxitool command only lists keys the CryptoServer's internal key database, <code>CXIKEY.db</code> .
<keystoreparam>	Configuration parameter for an external keystore If <code>KeyStoreType</code> is used, <code>KeyStoreParam</code> must be used as well. If <code>KeyStoreType</code> is used, it must be used before <code>KeyStoreParam</code> . <ul style="list-style-type: none"> <li>If <code>KeyStoreType=SDB</code> <code>KeyStoreParam</code> specifies the path to an external keystore file ( <code>/path/to/store.sdb</code> ).</li> <li>If <code>KeyStoreType=ODB</code> <code>KeyStoreParam</code> specifies the data source name as defined in format <code>DSN=DATA SOURCE NAME</code> or the complete ODBC connection string.</li> </ul>
<filename>	Path and name of the <code>.kbk</code> file the cryptographic key is restored from

<b>Example</b>	<code>cxitool Dev=3001@127.0.0.1 LogonPass=KeyMgrSlot0,12345678 Group=SL0T_0000 Spec=24 RestoreKey=C: \Utimaco\CryptoServer\SL0T_0000_24.kbk</code>
----------------	---

<b>Output</b>	Upon successful execution of the command, no output is given.
---------------	---

### 3.3.3.3 Restore Key with p11tool2 RestoreInternalKeys

This command restores all keys from the given key backup file to the internal key store.

<b>Syntax</b>	<code>p11tool2 [Slot=&lt;slot_id&gt;] &lt;login_key_manager&gt; RestoreInternalKeys=&lt;filename&gt;</code>
---------------	---



<b>Parameter</b>	<b>Description</b>
<slot_id>	ID of the slot as number Default: 0
<login_key_manager>	Login as key manager (via Login). Note: By default the key manager and the key user have the same permission mask. In that case the normal user can also be logged in (via LoginUser).
<filename>	Name of a previously generated key backup file

<b>Example</b>	p11tool2 Slot=1 Login=keyM,ask RestoreInternalKeys=C:/backup/internal_keys.bak
----------------	--

<b>Output</b>	16 internal keys restored to internal key store
---------------	---

## 4 References

<i><b>Document Number</b></i>	<i><b>Title/Company</b></i>
M010-0001	CryptoServer - Administration Manual / Utimaco IS GmbH.
2021-0006	u.trust Anchor LAN V5 - Administration Manual / Utimaco IS GmbH.
2009-0003	CryptoServer - csadm Manual / Utimaco IS GmbH.
2021-0037	u.trust Anchor - csadm Manual / Utimaco IS GmbH.
M013-0001-en	CryptoServer – PKCS#11 P11CAT Manual / Utimaco IS GmbH.
2012-0004	CryptoServer - PKCS#11 p11tool2 - Reference Manual / Utimaco IS GmbH.
2012-0007	CryptoServer PKCS#11 R3 - Developer Guide / Utimaco IS GmbH.
2020-0040	u.trust Anchor - Containerized Hardware Security Module (cHSM) - Administration Manual / Utimaco IS GmbH.
2020-0035	u.trust Anchor - Administration Manual / Utimaco IS GmbH.
2021-0072	u.trust Anchor - PKCS#11 p11tool2 - Reference Manual / Utimaco IS GmbH.

## 5 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](mailto: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.