

u.trust Anchor FIPS 140-3

Containerized Hardware Security Module (cHSM)

User 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	1.0.4
Product Version	6.0.0
Date	2024-10-25
Document No.	2023-0025
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	6
1.1	Target Audience for this Manual	6
1.2	Document Conventions	7
1.3	Abbreviations	7
2	Concepts.....	10
2.1	The Concept of the u.trust Anchor cHSM	10
2.1.1	Firmware Packages	10
2.1.2	FIPS Validated Firmware Package	11
2.2	Data Concepts and Definitions	12
2.3	cHSM States and Modes	15
2.3.1	cHSM Operating States	15
2.3.2	cHSM Operating Modes.....	15
2.4	Administration Tools	16
2.4.1	CryptoServer Command Line Tool csadm	16
2.4.1.1	csadm Commands for Cryptographic Users	17
2.4.2	Configuration and Key Management Services PKCS#11 Tool p11tool2.....	19
2.4.2.1	p11toolv2 Commands	20
2.5	Users and Authentication	24
2.5.1	Roles for cHSM Users.....	24
2.5.1.1	Cryptographic Users	24
2.5.2	Authentication Mechanisms.....	28
2.5.3	Permissions and Authentication Status	32
2.5.4	Maximum for Failed Authentication Attempts	35
2.6	Secure Messaging	35
2.7	External Interfaces	36
2.8	Key Usage and RSA Padding for FIPS cHSMs	37
2.9	CryptoServer APIs	38
3	Cryptographic Services	40
3.1	Administrative Functions	40
3.1.1	Get Info.....	41
3.1.2	Add User.....	41
3.1.3	Delete User	41
3.1.4	P11 Permissions.....	41

3.2	Key Management Functions	42
3.2.1	Initialize Key Group	42
3.2.2	Generate DSA Parameter	42
3.2.3	Generate DSA Parameter PQ	42
3.2.4	Generate DSA Parameter G	43
3.2.5	Backup Key	43
3.2.6	Restore Key	43
3.2.7	List Keys	44
3.2.8	Generate Key	45
3.2.9	Generate Key Pair	47
3.2.10	Open Key	47
3.2.11	Delete Key	47
3.2.12	Get Key Property	48
3.2.13	Set Key Property	48
3.2.14	Create Object	48
3.2.15	Copy Object	49
3.2.16	Derive Key	49
3.2.17	Export Key	50
3.2.18	Import Key	50
3.2.19	Split Key	51
3.2.20	Wrap Key	52
3.2.21	Unwrap Key	52
3.3	Cryptographic Functions	53
3.3.1	Compute Hash	53
3.3.2	Crypt Data	54
3.3.3	Sign Data	55
3.3.4	Verify Signature	55
3.3.5	Generate Random Number	56
3.3.6	Agree Secret	56
4	Troubleshooting	57
4.1	Alarm Treatment	57
4.2	Check Operativeness and State of cHSM	57
5	References	60

6 **Contact Address for Support Queries62**

1 Introduction

This manual provides information on the cryptographic usage of Utimaco's u.trust Anchor Containerized Hardware Security Modules (cHSM) running on a u.trust Anchor device.

All u.trust Anchor cHSMs share the same basic functionality. Depending on the template used for its creation, a cHSM can have additional functionalities, like complying with the approved mode of operation according to [FIPS140-3], for example. Special modes and behaviors for different cHSM functionalities are pointed out in this manual where they apply.

1.1 Target Audience for this Manual

This manual should be read carefully by all persons using the cryptographic services of the cHSM. u.trust Anchor cHSMs have a well-defined external software interface offering administration, configuration and cryptographic services. Only after an appropriate authentication is a user allowed to use the designated services.

In FIPS context, this manual should primarily be read carefully by all persons that assume the role of a **Cryptographic User (Key Manager or User)**. It does furthermore hold additional information for **Administrators** and **Security Officers**.

User Group	cHSM Configuration	Cryptographic Token or Slots Configuration	Key Management Services	Cryptographic Services	Set 'TRUSTED' attribute of Wrapping Key
Administrator	✓	✗	✗	✗	✗
Cryptographic User	✗	✗	✓	✓	✗
Key Manager	✗	✗	✓	✗	✗
User	✗	✗	✗	✓	✗
Security Officer	✗	✓	✗	✗	✓

Table 1: Permissions across User Groups



The security-relevant administrative commands that are offered by the u.trust Anchor cHSM, e. g. for loading, replacing, or deleting firmware, or for user management, are not described in this manual, since these administrative services can only be performed by persons who are allowed to assume the role of Administrator or Security Officer. The respective commands are described in the [u.trust Anchor - Administration Manual](#).

1.2 Document Conventions

We use the following document conventions:

<i>Convention</i>	<i>Use</i>	<i>Example</i>
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 2: 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.

1.3 Abbreviations

We use the following abbreviations in this manual:

Abbreviation	Description
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
GAAK	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

<i>Abbreviation</i>	<i>Description</i>
RSA	Rivest, Shamir, Adleman (cryptosystem)
SDMK	Sticky Device Master Key
TRNG	True random number generator

Table 3: Abbreviations

2 Concepts

This chapter describes background information of the software architecture and security mechanisms of the cHSM.

2.1 The Concept of the u.trust Anchor cHSM

The u.trust Anchor cHSM is a virtual, protected security module running on a u.trust Anchor device. It is developed specifically to ensure the efficient and secure performance of the following cryptographic operations:

- Generating keys
- Saving keys securely
- Generating random numbers (hardware (true) and software (pseudo) random number generator)
- Generating and verifying signatures
- Calculating hash values

The u.trust Anchor cHSM protects all cryptographic operations and used keys against any form of attack utilizing technical software solutions. The u.trust Anchor cHSM guarantees the trustworthiness and integrity of data within your IT systems.

In addition to the described cHSM functionality, the possibility to run FIPS cHSMs is provided. The u.trust Anchor is realized as multi-chip embedded cryptographic module, meeting FIPS 140-3 security level 3. FIPS cHSMs provide secure cryptographic services like encryption or decryption (for various cryptographic algorithms like RSA), hashing, signing, and verification of data (RSA, ECDSA), random number generation, on-board secure key generation, keystore, and further key management functions in a tamper-protected environment. FIPS cHSMs offer a general-purpose API with FIPS-approved algorithms for the above-mentioned cryptographic services.

2.1.1 Firmware Packages

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.

2.1.2 FIPS Validated Firmware Package

The following firmware modules are mandatory for the full functionality of FIPS cHSMs. They are loaded into the cHSM upon its creation by the u.trust Anchor administrator.

Module	File
ADM	adm.msc
AES	crypt.msc
ASN1	crypt.msc
DSA	crypt.msc
ECA	crypt.msc
ECDSA	crypt.msc
HASH	crypt.msc
LNA	crypt.msc
VRSA	crypt.msc
VDES	crypt.msc
CMDS	cmds.msc
CXI	cxi.msc
DB	db.msc
FIPS140	fips140.msc
HCE	hce.msc
POST	post.msc
MBK	mbk.msc
SMOS	smos.msc
UTIL	util.msc
CRYPT	crypt.msc

Table 4: Mandatory firmware modules for FIPS cHSMs

2.2 Data Concepts and Definitions

This chapter provides some important knowledge about the cryptographic concept of the CXI firmware module. Furthermore, it explains the main CXI definitions the understanding of which is essential for developing custom applications that shall use the cryptographic services offered by the CXI firmware module of the cHSM.

The generic data type used by the cryptographic CXI command interface of the cHSM is called CXI objects. A CXI object denotes a key, a configuration object or a storage object:

- A **Key** is a cryptographic key, key pair or generic secret to be used with a specific cryptographic algorithm (AES, RSA, ECDSA/ECDH, Generic Secrets to be used for HMAC calculation),
- A **Storage Object** may contain a certificate, domain parameters or other data that shall be stored in the cHSM's key table. Storage objects can only be used to store information securely within the cHSM; they cannot be used for any evaluation or calculation within the cHSM.
- A **Configuration Object** contains a list of configuration properties.



For a detailed description of the configuration properties and their usage, see section *SetConfig* in *CryptoServer – cxitool Manual* and section *SetGlobalConfig* in *CryptoServer - PKCS#11 p11tool2 - Reference Manual*.

The cryptographic services `OpenKey`, `ListKeys`, `GetKeyProperty`, `SetKeyProperty`, `DeleteKey`, `InitKeyGroup`, `BackupKey` and `RestoreKey` can be used with types of CXI objects.

The services `CreateObject` and `CopyObject` can only be used with keys and on storage objects, not on configuration objects.

All other services listed in this section can only be used with keys, neither with configuration objects, nor with storage objects.

Key Groups

Each Security Officer, Key Manager and User (and Cryptographic User) as well as every CXI object may be assigned to a key group:

- A CXI object is called **Global**, if it is not assigned to any key group; A CXI object is called **Local**, if it is assigned to a key group. In general, Local CXI objects can only be accessed by users who are assigned to the same key group (`CXI_GROUP`); Global objects can be seen by all users.
- A key or storage object is called **Assigned** to a user, if both belong to the same key group (`CXI_GROUP`), or if the CXI object is Global.
- Operators may be assigned to multiple key groups by using wildcards '*' and '?' within the `CXI_GROUP` user attribute.
- Key Managers and Users can only work with keys and storage objects.
- Security Officers can only access and edit Local configuration objects (configuration object whose key group matches the Security Officer's key group), and they can set the `CXI_PROP_TRUSTED` property of every Assigned wrapping key.
- Administrators are responsible for the Global configuration object.
- A Local configuration object only applies to the assigned key group; the properties of the Global configuration object apply to all Global objects, and to all Local objects with no Local configuration value defined.
The `CXI_PROP_CFG_ALLOW_GROUPS` (`CKA_CFG_ALLOW_SLOTS` in PKCS#11 API) property only exists in the Global configuration object; it does not exist in a Local configuration object.

Key Handle

A **key handle** denotes a hash value over the key group, key name and key specifier, and is needed to identify a CXI object in succeeding commands.

- The key group, key name and key specifier must be unique within the cHSM.
- Configuration objects are identified by the key group. They have neither a key name nor a key specifier. There exists exactly one Global configuration object. For each key group there exists exactly one Local configuration object.
- All keys and storage objects must be assigned to a key name and/or to a key specifier.

Key Blob

A key blob encapsulates a key or another CXI object and may contain a public and private key or a secret key. On FIPS cHSMs, private and secret key parts within a key blob must always be

encrypted.

The following types of key blobs are defined:

- Simple blob, MS CNG blob ("Bcrypt") and MS CSP blob ("Legacy"): Encapsulate keys and may be used as input or output parameter for the Export Key and Import Key function. They encapsulate the key components and additional parameters in different formats; the secret and private key components are encrypted with another Key (key encryption key).
- PKCS#11 blob: Encapsulates keys and is used as input or output parameter for the Wrap Key and Unwrap Key function. It encapsulates the key components and additional parameters as required by PKCS#11; the secret and private key components are encrypted with another Key (key encryption key).
- Backup blob: This format is used to handle CXI object backups and external keys. It contains an encapsulated property list, the key components (keys only) and a check value (a MAC calculated with the cHSM's MBK). The secret and private key components are encrypted with the cHSM's MBK.

Access Rules for Cryptographic Services

The following access rules are defined for the cryptographic services listed in the next sections:

- Users can execute the services ListKeys, OpenKey, GetKeyProperty, GenerateDSAParam, GenerateRandom, Crypt, Sign, Verify, ComputeHash and AgreeSecret.
- Key Managers can execute all listed services except for InitKeyGroup, GenerateRandom, Crypt, Sign and Verify.
- By configuration, both user roles Key Manager and User may be grouped together to one user role called Cryptographic User (this is the default configuration). Cryptographic Users can assume the Key Manager and the User role: They can execute all Key Manager and all User services, i.e. all listed services except for InitKeyGroup.
- Administrators can execute services on (Global) Configuration Objects: ListKeys, OpenKey, GetKeyProperty, SetKeyProperty BackupKey, RestoreKey, and DeleteKey. OpenKey and GetKeyProperty work on Local and Global Configuration Objects.

- Security Officers can execute services on Local Configuration Objects: ListKeys, OpenKey, GetKeyProp, SetKeyProp, BackupKey, RestoreKey, DeleteKey, and InitKeyGroup. InitKeyGroup may delete all Assigned Local CXI objects. Additionally, Security Officers may access any CXI object with the services OpenKey and GetKeyProp, and they can use the SetKeyProp service on keys to set the `CXI_PROP_TRUSTED` key property.

2.3 cHSM States and Modes

During its operation, the u.trust Anchor cHSM can enter several modes and operating states. The term "state" refers to the general state of operation of the cHSM, while the "mode" refers to additional restrictions that can apply.

2.3.1 cHSM Operating States

In error-free operation, a u.trust Anchor cHSM is always in state INITIALIZED. It can be used by the cHSM Administrator and assigned users.

Additionally for FIPS cHSMs, if the firmware module CRYPT fails to start, a u.trust Anchor FIPS cHSM can enter the state ERROR state.

```
mode      = Operational Mode
state     = INITIALIZED (0x00100004)
error state (<error code>)
```

To leave the ERROR state, restart the u.trust Anchor cHSM with the `csadm Restart` command. If this does not resolve the issue, please contact your Global Administrator. In state ERROR, many commands are blocked.

Information about the u.trust Anchor cHSM's state can be retrieved via the `csadm GetState` command, see *GetState* in the *u.trust Anchor - csadm Manual*.

See *Availability of commands on FIPS cHSMs* in *u.trust Anchor - csadm Manual* for a list of commands available in state ERROR.

2.3.2 cHSM Operating Modes

A functioning u.trust Anchor cHSM is always in Operational Mode. FIPS cHSMs can additionally be in a FIPS error state.

The current mode of the u.trust Anchor cHSM can be retrieved with the `csadm GetState` command. If the u.trust Anchor cHSM does not answer to the `csadm GetState` command, it is in Power Down Mode. In all other modes, the `csadm GetState` command can be performed.

Mode	Description
Operational Mode (optionally Administration-Only)	The u.trust Anchor cHSM is in Operational Mode if its operating system SMOS Façade and further firmware modules were started successfully and are active. The u.trust Anchor cHSM can be explicitly configured to boot into restricted Operational Mode (Operational Mode – Administration-Only). In this special kind of Operational Mode, all cryptographic functions are blocked and only administration functions can be executed, see <i>SetAdminMode</i> in the <i>u.trust Anchor - csadm Manual</i> . If the u.trust Anchor cHSM is not set to boot into Administration-Only Mode via the <code>csadm SetStartupMode</code> command, restarting the u.trust Anchor cHSM will enable full Operational Mode again.
Operational Mode - Administration-Only	The u.trust Anchor cHSM can be explicitly configured to boot into restricted Operational Mode. In this special kind of Operational Mode, all cryptographic functions are blocked, and only administration functions can be executed. A cHSM that is in Operational Mode – Administration-Only can be set back to the standard Operational Mode again, and vice versa. Standard Operational Mode means that the cryptographic functions are no longer blocked.
Power Down Mode	The cHSM is in power down mode if no firmware is active (cHSM is shut down). In power down mode, the cHSM is not able to receive any commands.

Table 5: cHSM Operation Modes

2.4 Administration Tools

The u.trust Anchor cHSM can be administered with two command-line tools, the CryptoServer administration tool `csadm` and the PKCS#11 Administration Tool `p11tool2`.

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

2.4.1 CryptoServer Command Line Tool `csadm`

The u.trust Anchor cHSM is administered with the CryptoServer command line tool `csadm`.



This manual describes the setup of these tools and the functionality of commands included in specific procedures to administer the cHSM. For a complete overview of all csadm commands and their parameters, refer to *u.trust Anchor - csadm Manual* within the product bundle.

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.

2.4.1.1 csadm Commands for Cryptographic Users

The following list gives an overview about all csadm commands which are available to a Cryptographic User. For a detailed description of all csadm commands, command syntax and authentication, see *u.trust Anchor - csadm Manual*.

Command	Description
Help	If called without any parameter, this command shows a list of all available csadm commands If the command name is given as a parameter, specific help will be provided
PrintError	Displays the corresponding error message text to an error code
Version	Shows the version of the csadm

cHSM Driver Commands

Command	Description
Reset	Performs a hardware reset (like Power Off-On) of the cHSM
Restart	Resets/restarts the cHSM and get it into operational mode
GetInfo	Retrieves information from the PCIe driver of the u.trust Anchor device
SetTimeout	Changes the maximum time that the driver waits for a response of the cHSM

Commands for cHSM Administration

Command	Description
GetState	Returns the status and mode of the cHSM
GetStartupMode	Outputs the operating mode of the cHSM after restart – either Operational Mode or Operational Mode – Administration-Only
StartOS	Starts the regular set of firmware modules (*.msc)
RecoverOS	Starts the recovery set of firmware modules (*.sys)
ListFiles	Lists all files stored on the cHSM
GetTime	Returns the cHSM's system time
ListFirmware	Returns a list with information about all firmware modules which are currently active
GetBootLog	Retrieves a log file which contains log messages made during the cHSMs boot process
GetAuditConfig	Displays the configuration for generation of the audit logfile
GetAuditLog	Displays the content of the audit logfile
Test	Executes a communication test with the cHSM

Commands for User Management

Command Authentication

Command	Description
GetCertificates	This command initiates a Secure Messaging handshake with the cHSM. It saves the der-encoded certificates sent by the device to the given directory
LogonSign	A user with RSA Signature authentication mechanism opens an authenticated Secure Messaging session for the given command
LogonPass	A user with HMAC Password authentication mechanism opens an authenticated Secure Messaging session for the given command
ShowAuthState	Displays the current authentication status on the cHSM
AuthHMACPwd	A user with the authentication mechanism 'HMAC Password' authenticates a single command

Miscellaneous Commands

Command	Description
Cmd	Provides a generic command interface
CmdFile	Provides a generic command interface Unlike the Cmd command it reads the input data from a file
Sleep	Delays the further command processing by the time given

2.4.2 Configuration and Key Management Services PKCS#11 Tool p11tool2

The *PKCS#11 Administration Tool p11tool2* (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 p11tool2 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 p11tool2 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 p11tool2 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 p11tool2 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 p11tool2 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 p11tool2.

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.

2.4.2.1 p11toolv2 Commands

This section gives an overview about all p11toolv2 commands which are available to the several user roles. See *CryptoServer - PKCS#11 p11tool2 - Reference Manual* for a complete list and a detailed description of all commands and installation procedure).

The authentication information is given as follows:

None: no authentication required

Any: valid authentication for any user role required

SO: Security Officer

CU/U: Cryptographic User or User

KM: Key Manager

Basic Commands

Command	Description	Authenticated by
Help	If called without any parameter, this command shows a list of all available p11toolv2 commands. If the command name is given as a parameter, specific help will be provided.	
PrintError	Displays the corresponding error message text to an error code.	
Version	Shows the version of the p11toolv2.	
ListSlots	Displays a list of all slots in the system.	
GetInfo	Displays general information about the cHSM PKCS#11 library.	
GetSlotInfo	Displays information about a specific slot.	
GetTokenInfo	Displays information about a specific cHSM.	
ListConfig	Displays a list of all configuration attributes.	
ListLocalConfig	Displays the value of the local (host) configuration attribute.	
GetBackupInfo	Displays information about a given backup file.	
LoginUser	Logs a standard PKCS#11 slot user (USER_<nnnn>) into the cHSM.	CU/U KM
Login	Logs any operator into the cHSM.	Any
SetPIN	Changes the password of the PKCS#11 standard slot Security Officer, (Cryptographic) User or Key Manager.	SO CU/U/KM
ListObjects	Displays a list of available keys and storage objects.	SO/CU/U/KM None
GetGlobalConfig	Displays the value of the global configuration attribute with the given name.	Any
GetSlotConfig	Displays the value of a slot configuration attribute.	SO CU/U/KM

General Commands

Command	Description	Authenticated by
LoginUser	Logs a standard PKCS#11 slot user (USER_<nnnn>) into the cHSM	CU/U KM
Login	Logs any operator into the cHSM	Any
SetPIN	Changes the password of the PKCS#11 standard slot Security Officer, (Cryptographic) User or Key Manager	SO CU/U/KM
ListObjects	Displays a list of available keys and storage objects	SO/CU/U/KM None
GetGlobalConfig	Displays the value of the global configuration attribute with the given name	Any
GetSlotConfig	Displays the value of a slot configuration attribute	SO CU/U/KM

Administrator Commands

Valid Administrator authentication required.

Command	Description	Authenticated by
InitToken (first execution per slot/group)	Initializes a slot by generating the PKCS#11 standard slot Security Officer	Administrator
SetGlobalConfig	Sets the value of a global configuration attribute	Administrator
DeleteSO	Deletes the the PKCS#11 standard slot Security Officer	Administrator

Security Officer (SO) Commands

Valid Security Officer authentication required.

Command	Description	Authenticated by
Login SO	Logs the standard PKCS#11 Security Officer (SO) into the cHSM	SO
InitToken (repeated execution per slot/group)	Re-initializes a slot by deleting all slot users and data (keys, storage objects and configuration settings)	SO
InitPIN	Generates the standard PKCS#11 User in a specific slot (group) with a given password	SO
SetSlotConfig	Sets the value of a slot configuration attribute	SO
BackupConfig	Creates a backup of the slot configuration object	SO
RestoreConfig	Restores the slot configuration object from the given configuration backup file	SO

Key Management Commands

Valid Key Manager authentication required.

Command	Description	Authenticated by
DeleteObjects	Deletes specified keys or storage objects	CU/KM
ImportP12	Imports an X509 certificate, a public or a private key	CU/KM
ImportCert	Imports an X509 certificate and a public key	CU/KM
ExportCert	Exports a certificate	CU/KM
GenerateKeyPair	Generates a public/private key pair	CU/KM
GenerateKey	Generates a secret key or set of domain parameters	CU/KM
BackupInternalKeys	Creates a backup of all available internal keys within a slot (see info below)	CU/KM
BackupExternalKeys	Creates a backup of all available external keys within the slot (see info below)	CU/KM
RestoreInternalKeys	Restores all keys from the given key backup file to the internal4 key store	CU/KM
RestoreExternalKeys	Restores all keys from the given key backup file to the external4 key store	CU/KM



Internal keys are stored within the cHSM; external keys are stored in a key database not located within the cHSM, e.g. on the host.

The number of cryptographic keys that can be stored inside the cHSM is limited by the storage capacity of the cHSM and depends on the key size and on the number and the size of the corresponding key properties. For example, either approximately

- 3000 2048-bit RSA keys or
- 14000 ECDSA keys (NIST-P256)

can be stored in the internal keystore of the cHSM. If you store 2048-bit RSA keys and ECDSA keys (NIST-P256) at the same time, the maximum number of keys to be stored is between 3000 and 14000. The number for the 2048-bit RSA keys stated above applies if not only a private key but also the corresponding public key is stored. This is the usual case. If PKCS#11 is used, the private key and the public key are stored in separate objects. If however only private keys are stored, either 4000 2048-bit RSA keys or 14000 ECDSA keys (NIST-P256) can be stored.

2.5 Users and Authentication

Certain external commands that are sent to the cHSM may only be executed if the sender has authenticated the command and a certain authentication state has been reached. For this purpose, one or more authentication header data blocks can be added to the command data block. These authentication headers will be processed completely by the firmware module CMDSD (command scheduler module). The addressed firmware module which will only receive the command data block is then responsible for checking the authentication state, if necessary, and to decide about its further execution.

Successful authentication can only be done by certain authorized users, which must have been registered previously in the cHSM. For this purpose, the cHSM administrates an internal user database. For a detailed list of the predefined user roles and their permissions, see section [Roles for cHSM Users](#).

Moreover, every user has the choice between two mechanisms of authentication: either via password (protected against being eavesdropped by a HMAC algorithm) or via RSA signature. The following subsections will explain the authentication mechanisms and usage in detail.

2.5.1 Roles for cHSM Users

2.5.1.1 Cryptographic Users

Additionally, you can create users assuming several roles, e. g. both the Administrator and User role, i. e. with user permission '22000002' (or higher). The cHSM allows users assuming different roles to log in simultaneously.

Users who can assume both the User and the Key Manager role are called *Cryptographic Users*. By default, the *Key Manager* permission and the *User* permission are the same (00000002) which means that the User role, the Key Manager role and the Cryptographic User role are identical. If the *Key Manager* permission mask is configured to 00000020, the Cryptographic User role is split into the two sub-roles Key Manager and User. Cryptographic Users must then have the permission 00000022 to be able to perform all cryptographic and key management services.

If you create any users with permissions in user groups different than the ones described above, they will not be allowed to authenticate and perform any command on a FIPS cHSM: Only permissions in the user groups 7, 6, 5, 2, 1 and 0 are relevant for authenticating commands on FIPS cHSMs. For example, a user who has been granted the 00010000 permission is not allowed to authenticate any command on a FIPS cHSM.

2.5.1.1.1 Tasks for Cryptographic Users

Cryptographic Users can perform the following tasks:

- **Execute administrative services using the administration tool csadm**

The Administration Tool csadm provides any kind of basic administration, e.g. download or deletion, retrieving status information, setting of the cHSM's clock, and user management.

- Most of these commands can only be executed by an Administrator since they are security relevant and have to be authenticated respectively.
- Some of these commands do not have to be authenticated at all (e. g. for retrieving status information). These commands can be used by every operator.
- Every operator can change his own authentication data (password or RSA key).

The csadm tool is available for various operating systems (Linux, Windows, Solaris). In chapter [csadm Commands for Cryptographic Users](#), the usage of the csadm tool is explained, including a detailed description of most csadm commands that are available for a Cryptographic User.

- **Execute PKCS#11 typical key management services using the PKCS#11 Administration Tool p11toolv2**

The PKCS#11 Administration Tool p11toolv2 provides any kind of basic configuration management and key management.

- Some of these commands do not have to be authenticated at all (e. g. for retrieving status information). These commands can be performed by every operator.
- Configuration management (display and setting configuration properties) can only be executed by Administrators and Security Officers.
- Key Management commands like key generation or deletion, key export or import, and key backup or restore can only be performed by a Cryptographic User or Key Manager.
- The standard PKCS#11 users can change their password.

The p11toolv2 tool is available for various operating systems (Linux, Windows, Solaris). In the chapter [Data Concepts and Definitions](#), the usage of the p11toolv2 tool is explained, including a short description of all p11toolv2 commands that are available for a Cryptographic User.

- **Develop own applications that use cryptographic services of the provided CryptoServer APIs**

E.g. for standard cryptographic interfaces like PKCS#11, JCE, OpenSSL etc., see also section [CryptoServer APIs](#). An application on the host PC can use one of the CryptoServer APIs to execute cryptographic services on a cHSM. These libraries are available for different operating systems.



In order to execute cryptographic functions, the Cryptographic User must perform an authentication and therefore have an account on the cHSM consisting of a user name and his authentication token, such as a password or an RSA key. Accounts must be created by an Administrator or Security Officer.

Command Execution with the csadm Tool

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).

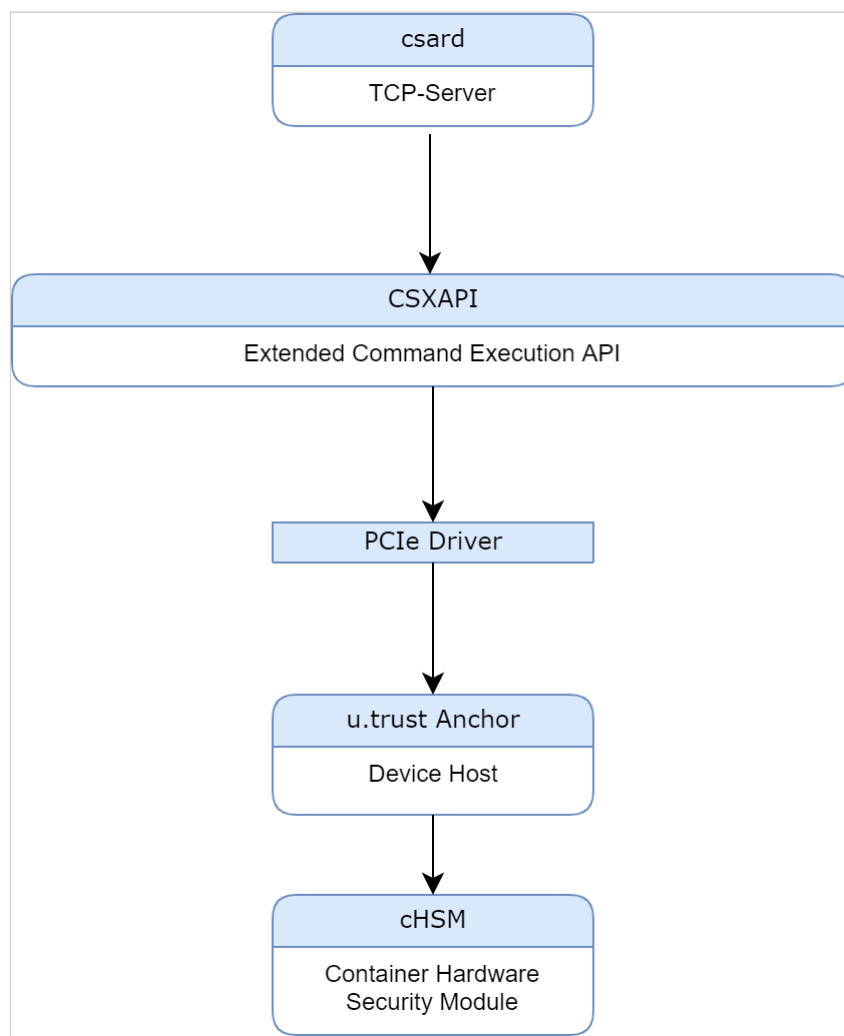


Figure 1 : u.trust Anchor csadm

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 6: Command classes

2.5.2 Authentication Mechanisms

The u.trust Anchor cHSM supports the following authentication mechanisms:

	HMAC Password Authentication	RSA Signature Authentication	ECDSA Signature Authentication
cHSM	✓	✓	✓
FIPS cHSM	✓	✓	✓

Table 7: Supported authentication mechanisms

HMAC Password Authentication

- **Description**

The u.trust Anchor cHSM supports the HMAC password mechanism for user authentication. The user's password may have a length between 8 and 1024 bytes.

The password is not transferred directly to the u.trust Anchor cHSM but is used as the input for the HMAC key-derived function (HMAC-PBKDF). A random number is added to each authentication. This prevents the authentication from being intercepted and performed again at a later point in time (replay attack).



Adding or restoring HMAC users with MD5 or Rd160 hash algorithms is not supported.

▪ Procedure

First, the host demands an 8-byte random value for each authentication from the cHSM. Then the host calculates the HMAC value over this random value and the command data block using the password as the input for the HMAC key-derived function (HMAC-PBKDF). This HMAC hash value is transferred to the cHSM together with the command data. The cHSM recalculates and checks the hash by using the password stored in the user database. The default hash algorithm for the HMAC calculation is SHA-256. Other hash algorithms can be used on demand for non-FIPS cHSMs.

The HMAC-PBKDF (HMAC password-based key-derived function) 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 fixed 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.

▪ Deployment

This authentication mechanism has the advantage that the password is not submitted in clear text and cannot be scanned. Because of the random value, the authentication data block can also not be scanned and replayed at a later time. A 'Playback'-attack of the command becomes impossible. Additionally, the command data is protected against unnoticed manipulation.

RSA Signature Authentication

▪ Description

The user's authentication token (public RSA key) is stored in the user database of the cHSM. Therefore, the user needs an RSA key pair either on a smartcard or in an optionally encrypted keyfile.



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.

The corresponding public part of the encryption key is stored as authentication data in the cHSM's user database `user.db`. If RSA authentication with a smartcard is used, the smartcard must be inserted into a PIN pad that is connected to a USB port of the computer on which the csadm tool is running.

▪ Procedure

First, the host demands an 8-byte random value from the cHSM. Then the host (or RSA smartcard) calculates an RSA signature over this random value and the command data block with the user's private RSA key (PKCS#11 signature format). This prevents a signed command from being intercepted and entered again later. The RSA signature will then be transmitted to the cHSM together with the command data block. The cHSM will verify the RSA signature with the help of the RSA key's public part stored in the user database `user.db`. The default hash algorithm for the signature calculation is SHA-256. Other hash algorithms can be used on-demand for non-FIPS cHSMs.

▪ Deployment

As RSA signature authentication does not transfer any confidential authentication data, it is particularly suitable for remote authentication to devices when the network connection is not completely under the control of the user.

ECDSA Signature Authentication

▪ Description

The user's authentication token (public ECDSA key) is stored in the user database of the cHSM. Therefore, the user needs an ECDSA key pair either on a smartcard or in an optionally encrypted keyfile.



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

The corresponding public part of the encryption key is stored as authentication data in the cHSM's user database `user.db`. If ECDSA authentication with a smartcard is used, the smartcard must be inserted into a PIN pad that is connected to a USB port of the computer on which the csadm tool is running.

▪ Procedure

For this mechanism, the host first demands an 8-byte random value from the cHSM. Then the host (or ECDSA smartcard) calculates an ECDSA signature over this random value and the command data block with the user's private ECDSA key. This command signature will then be transmitted to the CryptoServer which will verify it with the help of the public part of the ECDSA key which is stored in the user database `user.db`. The default hash algorithm for the signature calculation is SHA-256. Other hash algorithms can be used on demand for non-FIPS cHSMs.

▪ Deployment

As ECDSA signature authentication does not transfer any confidential authentication data, it is particularly suitable for remote authentication to devices when the network connection is not completely under the control of the user.

The following table shows which expressions are applied for using the authentication methods in some csadm commands.

<i>Authentication mechanism</i>	<i>csadm ListUser output value</i>	<i>csadm authentication commands</i>	<i>csadm user creation input parameter value</i>
HMAC password authentication	HMAC Password	csadm ... LogonPass=...	hmacpwd
RSA signature authentication with a keyfile	RSA Sign	csadm ... LogonSign=...	rsasign
RSA signature authentication with a smartcard			
ECDSA signature authentication with a keyfile	ECDSA Sign		ecdsla
ECDSA signature authentication with a smartcard			

Table 8: Naming of authentication mechanisms in csadm

The authentication mechanism that is applied to a user can be retrieved from the `Mechanism` column in the output of the `csadm ListUser` command.

Example

Name	Permission	Mechanism	Attributes
ADMIN	22000000	RSA sign	Z[0]
KeyUsrHmacPwd	00000002	HMAC	I[0]
KeyUsrRsaSignKeyfile	00000002	RSA sign	
KeyUsrRsaSignSmartcard	00000002	RSA sign	
KeyUsrEcdsaSignKeyFile	00000002	ECDSA sign	
KeyUsrEcdsaSignSmartcard	00000002	ECDSA sign	

The following table shows the identifiers of the authentication mechanisms. These identifiers are shown in some audit log entries, for example, when an authentication attempt has succeeded or failed.

<i>Authentication mechanism</i>	<i>Identifier</i>
HMAC password authentication	4
RSA signature authentication	0
ECDSA signature authentication	5

Table 9: Authentication mechanism identifier

2.5.3 Permissions and Authentication Status

The u.trust Anchor cHSM works with permission levels ranging from 0 (no permission/authentication) to 15 (highest level of permission/authentication).

<i>Permission</i>	<i>Meaning</i>
0	The user has no permissions in that particular user group.
1	Two-Person rule: although the user can authenticate commands to the cHSM, a second user is required for full authentication
2	The user is entitled to authenticate commands on their own.
3-15	Not required by the cHSM firmware functions by default. Can be required by functions with custom permissions defined in a signed configuration file <code>cmds.scf</code> .

Table 10: User permissions

Every user registered on the cHSM is assigned a permission level for each of the existing eight user groups. The permission level of the user defines which permissions can be reached effectively by one or more authentications, determining which commands the user is allowed to execute on their own, which commands need a second user for full authentication, and which commands cannot be performed at all. In communication with the

cHSM, the permission level defines which permission must be present before the cHSM can perform a particular command. After an authentication has been performed successfully by the user, the authentication status is increased by the user's permission, starting from authentication status 00000000.

Example for a User's Authentication Status

In this example, the cHSM has four users who have the following permissions.

User Group	7	6	5	4	3	2	1	0
ADMIN	2	2	0	0	0	0	0	0
Admin1	0	1	0	0	0	0	0	0
Admin2	0	1	0	0	0	0	0	0
User3	0	0	0	0	0	0	1	0

No user has been authenticated yet, so the authentication status is 00000000. To perform commands for user management, authentication status 2 has to be reached in user group 6. From the current authentication status 00000000, the following authentication scenarios are possible:

- **ADMIN authenticates**

The authentication status is updated to 22000000. The commands used for user management can now be performed because authentication status 2 has been reached in groups 6 and 7.

- **Admin1 authenticates**

The authentication status is updated to 01000000. The commands used for user management cannot be performed because authentication status 2 has not been reached in group 6.

- **Admin1 and Admin2 authenticate**

The authentication status is updated to 02000000. The commands used for user management can now be performed because authentication status 2 has been reached in group 6. The commands involved in user management cannot be performed because neither of the two administrators has the appropriate permission in group 7.

- **Admin1 and User3 authenticate**

The authentication status is updated to 01000010. Despite the fact that two authenticated users are now present, the commands used for user management cannot be performed because authentication status 2 has not been reached in group 6.



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 for Key Group Restrictions in Authentication Status

User Group	7	6	5	4	3	2	1	0
User1	0	0	0	0	0	0	0	1
User2	0	0	0	0	0	0	0	1

User1 has authentication status 0x00000001 and is assigned to the key group A.
User2 has authentication status 0x00000001 and is assigned to the key group B.

Three cryptographic keys are created.

Key	Key Group	Assigned User
KeyA	A	User1
KeyB	B	User2
KeyAB	AB	

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

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

Commands cannot be performed that require

- 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.

2.5.4 Maximum for Failed Authentication Attempts

Within the u.trust Anchor cHSM, 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 the u.trust Anchor cHSM administrator 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 u.trust Anchor cHSM 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 them. Consequently, this user cannot perform any command which must 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`.

2.6 Secure Messaging



For the EC-Diffie-Hellman key establishment protocol, see [\[SP80056A\]](#).

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).



A session key automatically becomes invalid if it is not used for 15 minutes. The cHSM 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.

2.7 External Interfaces



The Interface Specifications for all Firmware Modules is listed in the [References](#).

Some firmware modules (ADM, CMD5, CXI, MBK) offer functionality to the external world, i. e. callable from outside the cHSM. Such a command interface is called external interface.

A cHSM external interface expects command data coded together with a command header as a byte stream (via TCP). Then, such an external interface can be directly accessed by a host application or by one of the CryptoServer APIs that send/receive byte streams to/from the cHSM's internal interface. The functions building the external interface of a firmware module interpret the byte stream as a command like specified in the appropriate interface specification of the module. The answer of the module is also a byte stream, which is specified in the same document.

The different APIs which can be used by the host application generate a comfortable high-level interface out of the external cHSM byte stream interface. For this it composes the command byte stream from the different logical parts of the command header (for example, function code FC, sub function code SFC, command data length, etc.) and the block with the specific command data, and later decomposes the answer byte stream into the different

logical parts of the answer header and the block with the specific answer data. Then, this interface can be used by the host application: It hides the composition and decomposition of the command/answer header byte stream from the application programmer and therefore facilitates the usage of the cHSM protocol stack (the usage of the authentication layer). The task of the composition/decomposition and interpretation of the function specific command/answer data is left for the host application software. This must be done pursuant to the appropriate module specification.

Utimaco provides the following host software for cHSMs to access the external interface:

- For the external interface of the “administrative” firmware modules CMDS and ADM this task is done by the cHSM Administration Tool csadm.
- For the external interface to most administrative services of the CXI firmware module this task is done by the PKCS#11 Administration Tool p11tool2. This command-line tool provides PKCS#11-typical services to Administrators, Security Officers and Key Managers.

As interface to the cryptographic services, Utimaco provides various Cryptographic Services Interface Libraries. An application on the host computer can access the external byte stream interface directly or use one of these libraries to execute cryptographic services on a cHSM. Inside the cHSM, an external command of a firmware module is executed directly when it is called:



For external commands, there is only single command processing (“store and forward”) available. The commands will be performed one after the other, in that order in which they are received by the cHSM.

2.8 Key Usage and RSA Padding for FIPS cHSMs

On FIPS cHSMs, signature key pairs must not be used for any other purpose than signature generation and verification, see [FIPS186-4], chapter 3 “General Discussion”, page 11). For example, if an RSA key is used for key wrapping, it shall not be used for signature generation, and vice versa. Furthermore, on FIPS cHSMs, RSA signature keys must be used with a fixed “digital signature scheme” (ANS X9.31, RSASSA-PKCS1 v1.5 or RSASSA-PSS).

Both rules are enforced via the key property ‘FIPS usage’ (`CXI_PROP_FIPS_KEY_USAGE_PADDING`). This property enforces explicit checking of key usage and allowed padding mechanisms for any RSA, ECDSA or ECDH key. It is evaluated on FIPS cHSMs only, otherwise, it is ignored.

Key property FIPS Usage may take the following values for any RSA, ECDSA or ECDH key:

NO_SIGNATURE	FIPS key usage is anything but SIGN/VERIFY/VERIFY_RECOVER
SIGNATURE_v1_5	FIPS key usage is SIGN/VERIFY/VERIFY_RECOVER, and in case of an RSA key, padding mechanism is according to PKCS#1 RSASSA-PKCS1-v1_5
SIGNATURE_PSS	FIPS key usage is SIGN/VERIFY/VERIFY_RECOVER, and in case of an RSA key, padding mechanism is PKCS#1 RSASSA-PSS
SIGNATURE_ANSI	FIPS key usage is SIGN/VERIFY/VERIFY_RECOVER, and in case of an RSA key, padding mechanism is ANSI X9.31

Table 11: Values for key properties in FIPS usage

The value of the FIPS usage key property will be evaluated on any key usage operation, in addition to the evaluation of any other key properties. If the intended usage/padding does not fit the value, the command returns with an error code.

This property may be set on generation of the key or with the SetKeyProp function. If not yet set, it is automatically set on first usage (i.e. if an RSA key is first used as KEK for key wrapping, this property is set to NO_SIGNATURE, if an RSA key is first used for RSA signature generation with ANSI X9.31 padding, the property is set to SIGNATURE_ANSI). It may only be set once and cannot be changed afterwards.



A key may have contradicting properties, in this case all rules are enforced, which may lead to an unusable key. Example: If an RSA key is generated with key usage property set to SIGN/VERIFY, and simultaneously FIPS usage set to NO_SIGNATURE, the key can neither be used for signature generation/verification nor for anything else. The cHSM does not prevent this from happening.

2.9 CryptoServer APIs

Different Application Programming Interface (API) libraries are available for different cryptographic interfaces. With all of them it is possible either to access a cHSM over TCP/IP. All APIs require that the firmware module CXI is installed on the cHSM.

API Library	Description
CXI API (Java)	The CXI API provides a proprietary cryptographic Java interface for the cHSM. To use this interface, the application must have the CryptoServerCXI.jar library file in the Java classpath.
CXI API (C++)	The CXI API provides a proprietary cryptographic C++ interface for the cHSM. To use this interface, the cxi.h file must be included, and the application must either load the cxi.dll/so library file manually or must be linked against cxi.lib and be able to find the CXI cxi.dll/so library file in the path.

API Library	Description
JCE	JCE is a cryptographic extension to Sun's Java framework. It defines a unified interface between an application and a cryptographic device. With this concept a Java application must not know about specific drivers to access the cHSM directly. To use Utimaco's JCE provider the CryptoServer JCE provider (CryptoServer.JCE.jar) must be installed and configured on the host system.
EKM	Currently, Utimaco's EKM Provider supports Microsoft SQL Server 2012/2014/2016/2017 on Windows Server 2012 R2/2016/2019. To use Utimaco's EKM provider, cssqlekm.dll and cssqlekm.lib.dll must be installed and configured on the host system.
CryptoAPI/CSP	The Utimaco CryptoServer CSP implements a full RSA and AES provider. To use this provider the Utimaco's CSP library (cs2csp.dll and cs2cng.dll) must be installed and configured on the host computer
CNG	The Utimaco CryptoServer Key Storage Provider implements a CNG key storage interface. To use this provider, the Utimaco's CNG library (cs2cng.dll) must be installed and configured on the host computer.
PKCS#11	The PKCS#11 standard specifies an application programming interface (API), called "Cryptoki," for devices that hold cryptographic information and perform cryptographic functions. It is developed by OASIS Open [PKCS11]. The cHSM provides a PKCS#11 compliant interface. To use this interface, the application must either load Utimaco's CryptoServer PKCS#11 library (cs_pkcs11_R2.dll/so) manually or it must linked against cs_pkcs11_R2.lib and be able to find the library file in the path.

Table 12: CryptoServer API Libraries

Respective API libraries are available for various Windows and Linux operating systems. They can be found on the delivered product CD in the ...\\Software\\<OS>\\<OS distribution>\\Crypto_APIS\\<API>\\lib\\ folder.



CryptoAPI is a general-purpose cryptography standard developed by Microsoft. On the top side, it defines a cryptographic interface to be used by applications, on the bottom side, it defines an interface to be used by manufacturers for integrating their cryptographic hardware using a CSP (Cryptographic Service Provider). With this concept, the application must not know about specific drivers to access cryptographic hardware directly. Refer to Microsoft's MSDN web pages for a detailed specification of the CSP functionality.



CNG (Crypto Next Generation) is a new cryptographic interface, which has been introduced on Vista and Windows Server 2008. It offers updated cryptographic algorithms and is intended as long term replacement of CSP. For now, CSP is still supported on Vista and Windows Server 2008. Please refer to Microsoft's MSDN web pages for a detailed specification of the CNG functionality.

3 Cryptographic Services

This chapter gives an overview about the cryptographic services offered by the cHSM's CXI firmware module to a Cryptographic User for developing custom applications. Furthermore, most of the cryptographic and key management functions described here are offered by Utimaco's CXI tool (cxitool) which is a command-line utility used, for example, for generating cryptographic keys, encrypting, decrypting and signing data and verifying digital signatures. This tool is provided on the product CD for Linux operating systems. For details about tool installation, features and usage, read [\[CSCXI\]](#).

The CXI firmware module offers administrative, key management and cryptographic services to the external world by providing a byte stream interface called external CXI Interface. It can be directly accessed by a host application or by one of CryptoServer APIs that sends/ receives byte streams to/from the u.trust Anchor device.

Utimaco offers different libraries as interface to the cryptographic services. These libraries can be used by the host application to access cryptographic services in a comfortable way, see chapter [CryptoServer APIs](#). Although we highly recommend using these APIs, the cHSM can also be used without an API. In this case the application itself must create the command and parse the byte stream answers. For that purpose, the external functions of the CXI firmware module must be called by their unique subfunction code (SFC or function ID) inside the CXI module combined with the unique function code (FC or module ID) of the CXI module (0x68).

All cryptographic services can only be performed if they have been previously authenticated by a Cryptographic User within a Secure Messaging session.



For details about the individual CXI external functions and instructions on using them, see the HTML documentation provided within the product bundle in the ...
`\Documentation\Crypto_APIS\CXI_JAVA\html` folder.

3.1 Administrative Functions

This chapter lists the administrative functions provided by the CXI firmware module. They allow any user to get general information, for example, about the CXI module and CXI API version or enable an Administrator or a Security Officer to create/delete other CXI/PKCS#11 specific user roles. These functions can also be performed by using the cxitool (see [\[CSCXI\]](#)) except for the P11Permissions function.

3.1.1 Get Info

This is a general function offered by the CXI firmware module to all user roles (Administrators, Cryptographic Users, Key Managers and Users). It does not require user authentication and returns some information about the CXI firmware module (for example, version number of CXI firmware module and fill level of the key database).

3.1.2 Add User

For further details about the cHSM's user concept, see sections [Roles for cHSM Users](#) and [Users and Authentication](#).

Permission

On creation of a Security Officer, the function must be authenticated by an Administrator (permission 20000000).

On creation of a Key Manager or User, the function must be authenticated by a Security Officer (permission 00000200), who must be a member of the same key group (`CXI_GROUP=<group name>`) as the user to be created.

3.1.3 Delete User

For further details about the cHSM's user concept, see sections [Roles for cHSM Users](#) and [Users and Authentication](#).

Permission

On deletion of a Security Officer, the function must be authenticated by an Administrator (permission 20000000).

On deletion of a Key Manager or User, the function must be authenticated by a Security Officer (permission 00000200), who must be a member of the same key group (`CXI_GROUP=<group name>`) as the user to be deleted.

3.1.4 P11 Permissions

This function returns information about the user role (defined according to [PKC#11](#): Cryptographic User, Security Officer and Key Manager) of the users with CXI_GROUP attribute matching the specified Key Group, who are currently logged in to the cHSM.

The returned information is a 4-byte bit field.

<i>Bit</i>	<i>User Role</i>
0	Cryptographic User
1	Key Manager

Bit	User Role
2	Security Officer

Table 13: p11 permissions

If the required authentication level for the corresponding role has been reached, the corresponding bit in the field is set to 1. Otherwise, it is set to 0.

This function does not require user authentication.

3.2 Key Management Functions

This chapter lists the external key management functions of the CXI firmware module. Most of these commands can also be performed by using the `cxitool`, see the [CryptoServer Cryptographic Service Interface – Firmware Module CXI – Interface Specification](#).

3.2.1 Initialize Key Group

This function removes all CXI objects which belong to the given key group and which are assigned to the authenticated Security Officer from the internal key database of the cHSM. Input data is the respective key handle or key template optionally containing the key group.

This service is available to Security Officers only. There is no output data.

3.2.2 Generate DSA Parameter

This function generates DSA parameters p , q and g of given length. If the cHSM is a FIPS cHSM, only parameter lengths $|p| / |q| = 2048/224, 2048/256$ or $3072/256$ are allowed, and parameters are always generated according to [FIPS1864].

This function is available for (Cryptographic) Users and Key Managers.

3.2.3 Generate DSA Parameter PQ

This function generates DSA parameters p and q of given length. If the cHSM is a FIPS cHSM, only parameter lengths $|p| / |q| = 2048/224, 2048/256$ or $3072/256$ are allowed, and parameters are always generated according to [FIPS1864].

This function is available for (Cryptographic) Users and Key Managers.

3.2.4 Generate DSA Parameter G

This function generates DSA parameter g from given p and q . If the CHSM is a FIPS cHSM, only parameter lengths $|p| / |q| = 2048/224, 2048/256$ or $3072/256$ are allowed, and parameter is always generated according to FIPS 186-4.

This function is available for (Cryptographic) Users and Key Managers.

3.2.5 Backup Key

This function returns a backup blob which is encrypted with the cHSM's MBK.

Input parameter is the key handle of a CXI object stored in the internal key database. Output parameter is the requested backup blob containing:

- all assigned properties,
- for keys: the secret key (encrypted) or the public and private (encrypted) key parts,
- and a check value (an MBK based MAC over all preceding output parameters).

With this function:

- Key Managers can back up keys and Storage objects,
- Security Officers can back up Local Configuration Objects,
- Administrators can back up the Global Configuration Object.

3.2.6 Restore Key

This function imports a backup blob which is encrypted with the cHSM's MBK.

Input parameters are:

- command flags indicating the storage mode (existing CXI object shall be overwritten, CXI object shall not be stored within the cHSM but shall be returned as MBK encrypted,
- blob, key shall be stored permanently or not - in the latter case it will be deleted on restart of the cHSM),
- backed up CXI object to be restored (MBK encrypted backup blob, including the check value),
- and optionally a list of CXI object properties which shall be set for the restored CXI object.

If the decryption with the MBK was successful and the check value could be verified, the additional properties are added to the unwrapped key object, if given. The unwrapped CXI object is then either stored in the cHSM's internal key table under the respective key handle together with its properties, or an MBK encrypted backup blob is generated, if requested. Output parameter is the newly generated backup blob, if requested or otherwise, the key handle of the stored CXI object.

With this function:

- Key Managers can restore keys and Storage objects,
- Security Officers can restore Local configuration objects,
- Administrators can restore the Global configuration object.

3.2.7 List Keys

This function returns the property lists of all CXI Objects stored in the internal key database of the cHSM which match the given search criteria and which the requesting user is allowed to see after authenticating the function:

<i>User role</i>	<i>Permissions</i>	<i>Objects returned</i>
User	configurable (default: 00000002)	Keys and storage objects
Key Manager	configurable (default: 00000002)	Keys and storage objects
Security Officer	00000200	Local (group-specific) configuration objects
Administrator	20000000	Global configuration object

Table 14: User roles and accessible CXI objects

This list may also be empty, for example, if no operator is authenticated, or if no key group matches.

For every such CXI object the following data are output:

- Key handle,
- Key name (if set),
- Key group (if set),
- Key specifier (if set),

- Key algorithm (AES, RSA, ECDSA, ECDH, RAW, X509, X509_ATT),

- Key size (if set),
- Key type (public key, private key, secret key).

3.2.8 Generate Key

This function creates a new key (or key pair) according to the given property list. The generated key is either internally stored (and the corresponding key handle is returned) or the key is not stored in the key database but an MBK encrypted backup blob is returned.

This function is available for Key Managers only.

Input Parameter

The following properties must be given as input data:

- algorithm (AES, RSA, ECDSA, ECDH, RAW),

- key type,
 - key size in bits for AES, RSA, RAW.
For FIPS cHSMs, only the following key sizes are allowed:
 - AES: 128, 192 or 256 bits,
 - RSA: ≥ 2048 bits,
 - Generic Secret (Algorithm 'RAW'): ≥ 112 bits.

Further properties may be given as input data, for example:

- key group, name and specifier,
- export permission (allowed, allowed in plain, deny MBK encrypted backup),
- key usage (ENCRYPT, DECRYPT, SIGN, VERIFY, DERIVE, WRAP and UNWRAP),
- FIPS usage (NO_SIGNATURE; SIGNATURE_V1_5; SIGNATURE_PSS; SIGNATURE_ANSI)



A key may have contradicting key usage and FIPS usage properties, the cHSM does not check these for compatibility. In this case, all rules are enforced, which may lead to an unusable key. Example: If an RSA key is generated with key usage set to SIGN/VERIFY only, and simultaneously FIPS_USAGE set to NO_SIGNATURE, the key can neither be used for signature generation/verification nor for anything else.

- key label,
- key generation date,
- key certificate.

Input Flags

- CXI_FLAG_KEY_OVERWRITE (overwrite already existing key)
- CXI_FLAG_KEY_EXTERNAL (don't store key on cHSM but return MBK encrypted Backup Blob)
- CXI_FLAG_KEY_VOLATILE (don't store key permanently, i.e. key will be deleted on restart of cHSM)

Parameters depending on Algorithm

- **RSA**
 - use hardware (real) or deterministic (pseudo) random number generator,
 - public exponent (optional, default: 010001),
 - For FIPS cHSMs: key generation mode = RSA_MODE_FIPS_PRIME (according to [FIPS1864]),
 - length difference of primes p and q in bits (default 0).

- ECDSA, ECDH

- use hardware (real) or deterministic (pseudo) random number generator,
- public key format (compressed, uncompressed or hybrid form).

3.2.9 Generate Key Pair

This function generates a new asymmetric key pair similar to function Generate Key, see section [Generate Key](#). In difference to the Generate Key function, no symmetric algorithm can be given and the two key parts are stored in two different key objects (with different key handles): Two different property lists for each key part are given as input parameters, and two different key handles or backup blobs may be returned.

This function is available for Key Managers only.

3.2.10 Open Key

This function opens an existing CXI object and returns a key handle which allows identifying the CXI object in further commands, or a backup blob if requested. A template containing the name and optionally key group and specifier is given as input data.

With this function:

- Key Managers, Users and Security Officers can open all (Assigned) CXI objects, and
- Administrators can open all Configuration objects.

3.2.11 Delete Key

This function removes a CXI object from the internal key database of the cHSM. Input data is the respective key handle or a key template. There is no output data.

With this function:

- Key Managers can delete keys and storage objects,
- Security Officers can delete Local configuration objects, and
- Administrators can delete the Global configuration object.

3.2.12 Get Key Property

This function returns one or more properties of a given CXI object. Input data are:

- a CXI object (in form of a key handle or backup blob) whose properties are to be inquired,
- a list of properties to be inquired.

Output data is the filled property list containing the actual property values.

With this function:

- Key Managers, Users and Security Officers can access all (Assigned) CXI objects,
- Administrators can access all configuration objects.

3.2.13 Set Key Property

This function sets one or more properties for the given CXI object. Input data is the CXI object whose properties are to be set (as key handle or as backup blob), and a list of property values to be set.

Output data is the modified CXI object (backup blob if input was given as backup blob, key handle otherwise).

With this function:

- Key Managers and Users can update keys and storage objects,
- Security Officers can update Local configuration objects and the `CXI_PROP_TRUSTED` property of each key object,
- Administrators can update the Global configuration object.

3.2.14 Create Object

This function generates a new key or storage object in the database. All required properties must be given in the property list.

This function is available for Key Managers only.

3.2.15 Copy Object

This function copies a key object or a storage object. A key template may be given that contains a list of properties which should be added to the original properties or replace existing properties.

The copied object is either stored in the internal key database of the cHSM or is returned as an MBK encrypted backup blob.

This function is available for Key Managers only.

3.2.16 Derive Key

This function derives an AES key or a generic secret from a base key (AES or ECDH).

Input parameters are storage flags and property list for the derived key, the base key (as a key handle or backup blob), the key derivation function KDF and data as needed by chosen KDF (for example, chaining mode, initialization vector, data to be encrypted/hashed/XORed/appended/prepended, hash algorithm, public key, key to be concatenated (a key handle or a backup blob), offset size). The base key must have the key usage property DERIVE.

The derived key is either stored in the internal database or returned as an MBK encrypted backup blob.

For cHSMs,

- the base key must have the FIPS usage property NO_SIGNATURE,
- only KDFs ECDH, ECDH_COF and TLS12_PRF are allowed,
- TDES keys are blocked,
- RAW keys (base keys and derived keys) must have a size of at least 112 bits,
- EC curve of base keys must be Allowed or Approved as listed in Appendix A,
- Base key and SHA function must provide sufficient security strength for the derived key (see NIST-P 800-57 part1, Table 2+3),
- Keys derived with ECDH cannot be stored in the internal database.

This function is available for Key Managers only.

3.2.17 Export Key

This function exports a key from the internal key database of the cHSM.

Input parameter is

- the key handle or (MBK encrypted) backup blob to be exported,
- the key type to be exported: private/public/secret,
- the key blob format,
- optional for public keys to be exported, and optional for all keys with `CXI_KEY_EXPORT_ALLOW_PLAIN` property set to YES: the key encryption key (with key usage property `WRAP`) to be used to encrypt the secret key component and the chaining and padding mode.

For FIPS cHSMs,

- the key encryption key must have FIPS usage property `NO_SIGNATURE`,
- TDES keys are blocked,
- if AES GCM is used for key wrapping, the initial IV must be generated internally,
- RSA wrapping keys must have a key size of at least 2048 bits.

The function returns the key to be exported including key algorithm, key size and secret/private and/or public part (private or secret key encrypted with key encryption key) encoded as Key Blob in the requested format.

When used on internal keys, the key (pair) remains stored in the key database. The key to be exported must have the `CXI_KEY_EXPORT_ALLOW` property. The key encryption key must have the key usage property `WRAP`.

This function is available for Key Managers only.

3.2.18 Import Key

This function imports a key into the internal key table of the cHSM.

Input parameters are:

- the key type (public, private, secret),

- the key blob type,
- key attributes,
- key storage mode (volatile/non-volatile storage, overwrite/don't overwrite existing key with given key handle, do/do not store the key on the cHSM but return an MBK encrypted backup blob),
- optional: the key encryption key to be used to decrypt and the chaining and padding mode used to encrypt the private/secret key component (key usage property `UNWRAP`),
- and the key (optionally encrypted), including algorithm and key size encoded in a key blob.

The imported key and its attributes are stored in the cHSM's internal key database (if internal storage is wanted). In case of external storage, the MBK encrypted key is returned (backup blob format). In case of internal storage, a key handle is returned. The key encryption key (KEK) must have the key usage property `UNWRAP` .

For FIPS cHSMs,

- the key encryption key must have FIPS usage property `NO_SIGNATURE`,
- TDES keys are blocked,
- RSA wrapping keys must have a key size of at least 2048 bits.

This function is available for Key Managers only.

3.2.19 Split Key

This function splits a base key of type Generic Secret (RAW) into N non-overlapping segments which are stored as new keys of type DES, AES or Generic Secret according to the given flags, offsets and key templates (with $N \leq 15$).

The base key must be an internal key. It is deleted (and zeroized) after successful command execution. The resulting keys are either stored in the internal database or returned as an MBK encrypted backup blob.

For FIPS cHSMs, the following restrictions apply:

- Derived DES keys must have a key length of 112 or 168 bits (16 or 24 bytes),
- base and derived Generic Secrets must have a length of ≥ 112 bits.

The command has to be authenticated as Key Manager. The user has to be member of all key groups, meaning the base key and of all derived keys.

3.2.20 Wrap Key

This function exports a key from the internal key database of the cHSM encrypted with a key encryption key (KEK) and formatted according to [PKCS11]. The key to be exported must have the `CXI_KEY_EXPORT_ALLOW` property, the KEK must have the key usage property `WRAP`.

Data input parameters have to be the key IDs or backup blobs of both keys, padding and chaining mode, prefix and postfix data and Initialization Vector (optional; if wrapped key is AES).

For FIPS cHSMs,

- The key encryption key must have FIPS usage property `NO_SIGNATURE`,
- TDES keys are blocked,
- if AES GCM is used for key wrapping, the initial IV must be generated internally,
- RSA wrapping keys must have a key size of at least 2048 bits.

The function returns the wrapped key and its key type, key size, and key attributes.

This function is available for Key Managers only.

3.2.21 Unwrap Key

This function imports an encrypted key into the internal key table of the cHSM. The key encryption key (KEK) must have the key usage property `UNWRAP`.

Input parameters have to be the key handle or backup blob of the KEK, the wrapped key (as PKCS#11 blob), key properties to be set for the unwrapped key, and the wanted key storage mode.

For FIPS cHSMs,

- The key encryption key must have FIPS usage property NO_SIGNATURE,
- TDES keys are blocked,
- RSA wrapping keys must have a key size of at least 2048 bits.

This function is available for Key Managers only.

3.3 Cryptographic Functions

This chapter lists the external cryptographic functions of the CXI firmware module. They offer services for symmetric (DES, AES) and asymmetric (RSA, ECDH, ECDSA) cryptography, hashing and random number generation. All of these functions except for Generate Random Number and Agree Secret can also be performed by using the cxitool, see [\[CSCXI\]](#).

3.3.1 Compute Hash

This function computes a hash or HMAC value over given data or key components with a chosen algorithm.

Input parameters are the data to be hashed or HMACed, the hashing/HMAC algorithm, the HMAC key (optional), a flag if multipart hash calculation shall be done and (optionally) Hash digest information, returned on a previous call of this function, to be used to continue a multipart hash calculation.

For FIPS cHSMs,

- only the SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and SHA-3 hashing algorithms can be used,
- TDES keys are blocked,
- RAW HMAC keys (Generic Secrets) must have a length of ≥ 112 bits.

The data to be hashed may either be given in plain, or in the format as output by the Agree Secret function (this means MBK encrypted and with check value). Optionally, additional data may be input. If given, this data is prepended or appended to the data before hash calculation.

<p>The hash is calculated over the concatenation of the following items:</p> <ul style="list-style-type: none">▪ Data to be prepended▪ Data▪ Key components of secret key▪ Data to be appended	<p>The HMAC is calculated over concatenation of the following items:</p> <p>Data to be prepended</p> <p>Data</p> <p>Data to be appended</p>
---	---

Table 15: Hash calculation

Any missing item or item with a length of zero is skipped.

Output parameter is the calculated hash or HMAC value, or the Hash digest information to be used to continue a multipart hash calculation, if requested.

This function is available for (Cryptographic) Users and Key Managers.

3.3.2 Crypt Data

This function encrypts or decrypts data with a DES or an AES key in OFB, ECB or CBC mode.

Input parameters must be the following:

- the key given as key handle or backup blob (if data shall be decrypted, the key to be used must have the key usage property `CXI_MECH_MODE_DECRYPT` ; if data shall be encrypted, the key to be used must have the key usage attribute `CXI_MECH_MODE_CRYPT`),
- operation mode (encrypt or decrypt),
- chaining mode (OFB, ECB or CBC),
- OFB and CBC mode only: initialization vector (optional),
- padding mode and hash algorithm if required,
- data to be encrypted or decrypted.

For FIPS cHMs,

- the en/decrypton key must have the FIPS usage attribute `NO_SIGNATURE`,
- TDES keys are blocked,
- for AES GCM encryption, the initial IV must be generated internally,

- RSA and ECDSA keys are not allowed to be used for data decryption or encryption.

The command returns the encrypted or decrypted data (and, in OFB and CBC mode, the final chaining vector).

This function is available for (Cryptographic) Users only.

3.3.3 Sign Data

This function calculates one of the following:

- for asymmetric algorithms: a signature for a given hash value,
- for symmetric algorithms: a message authentication code (MAC).

The input parameters must be as follows:

- The signing or MAC key as key handle or backup blob (the key to be used must have the key usage property SIGN),
- Hashing algorithm,
- Padding algorithm and parameters,
- MAC calculation only: chaining mode and initialization vector to be used for chaining (optional),
- Hash value to be signed or data to be MACed.

The command returns the calculated signature or MAC. For large data to be MACed the data can be split into several blocks and the function can be called for each block separately. The calculated MAC must be used as IV for the MAC calculation of the next block.

This function is available for (Cryptographic) Users only.

3.3.4 Verify Signature

This function verifies a signature or MAC. The input parameters must be as follows:

- the signature key or MAC key as key handle or backup blob (the key must have the key usage property VERIFY),
- padding algorithm,

- hashing algorithm,
- mAC only: chaining mode and initialization vector to be used for chaining (optional),
- hash value and corresponding signature (if asymmetric key is given) or MACed data and corresponding MAC (if symmetric key is given),
- the signature (if asymmetric key is given) or MAC (if symmetric key is given) which shall be verified.

Successful completion of the command means that the signature has been verified successfully.

If chaining is used (for large MACed data the data can be split into several blocks and the function can be called for each block consequently) the input parameter 'corresponding MAC' is left empty (for all but the last chaining block) and the calculated 'preliminary' MAC is output in encrypted form which has to be used as (encrypted) initialization vector for the next chaining block.

This function is available for (Cryptographic) Users only.

3.3.5 Generate Random Number

This function generates a random number of arbitrary length. The Entropy source is compliant to [\[NIST SP 800-90B\]](#); the DRNG is based on the SHA-512 algorithm as transition function and compliant to [\[NIST SP 800-90A\]](#).

This function is available for (Cryptographic) Users only.

3.3.6 Agree Secret

This function derives a shared secret from a public and a private EC key agreement key (the key must have the key usage property DERIVE) according to [\[TR-03111\]](#), Section 4.3.1). The input parameters are the private part of key 1 and the public part of key 2 (given as a key handle or a backup blob) for the key agreement. The function returns the calculated shared secret encrypted with the cHSM's MBK, and an integrity check value.

This function is available for (Cryptographic) Users and Key Managers.

4 Troubleshooting

4.1 Alarm Treatment

An alarm can not be triggered on the cHSM itself, but will always occur on the u.trust Anchor device itself. If a cHSM is accessible, this means no alarm state is present on the u.trust Anchor device.

4.2 Check Operativeness and State of cHSM

This section deals with the situation where it is not known whether the cHSM works at all, and in which mode/state it is. The following steps should be systematically performed to check the cHSM's operativeness and, if possible, to get the cHSM working again.

In some cases, it will be necessary to ask a cHSM Administrator for help.

Precondition

- Make sure that the administration tool csadm is installed on the computer where the cHSM is running.
- Make sure that the cHSM and the host PC, whereon the administration tool csadm is installed, are properly connected to the network (try to 'ping' the cHSM from the host PC).

Procedure

1. Perform the `GetState` command, see [csadm Commands for Cryptographic Users](#).
In this table the operativeness check is done by considering the output data for the mode, state and alarm of the csadm `GetState` command.). Compare answers listed in the following table.

Result	Explanation/Reason/Adjustment
No answer	Dead state or power down mode → Restart the cHSM. → Contact the u.trust Anchor Global Administrator.
Error Code B9011xxx, B9015xxx, B9016xxx, B9017xxx or B9021xxx to B9024xxx	u.trust Anchor's PCIe carrier card does not react. → Try to restart the cHSM with the <code>csadm Restart</code> command. → Contact the u.trust Anchor Global Administrator.

Result	Explanation/Reason/Adjustment
Error Code B901xxxx or B902xxxx	No connection to the cHSM due to a communication problem, wrong host or device name, network problem. <ul style="list-style-type: none"> • Check the <code>csadm GetState</code> command syntax for the correct Dev parameter • Check the connection to the cHSM by performing a 'ping' command. • Check state/configuration of the TCP daemon on the cHSM.
mode = Operational Mode state = INITIALIZED error state (0xb0830025) alarm = OFF or mode = Operational Mode - Administration-Only state = INITIALIZED error state (0xb0830025) alarm = OFF	The cHSM is in an error state. → Contact the cHSM Administrator.
mode = Operational Mode - Administration-Only state = INITIALIZED alarm = OFF or mode = Operational Mode - Administration-Only state = INITIALIZED alarm = OFF	All cryptographic functions are blocked, and only administrative functions can be executed. → Contact the cHSM Administrator for help to switch back to full operational mode.
mode = Operational Mode state = INITIALIZED FIPS mode = ON alarm = OFF	The u.trust Anchor device is using the FIPS firmware package and the cHSM is using the SecurityServer FIPS template. It is fully operational.
mode = Operational Mode state = INITIALIZED FIPS restrictions = applied alarm = OFF	The cHSM is using the SecurityServer FIPS template. It is fully operational.
mode = Operational Mode state = INITIALIZED alarm = OFF	The cHSM is fully operational.

Table 16: Interpretation of return of GetState

2. Contact a cHSM administrator to restart the cHSM. Wait a minute, then perform a `GetState` command.

<i>Result</i>	<i>Explanation/Reason/Adjustment</i>
no error	The cHSM is operational.
any error	→ Contact the u.trust cHSM Administrator.

Table 17: Interpretation of return of GetState after restart

5 References

Reference	Title/Company	Document Number
	u.trust Anchor - Containerized Hardware Security Module (cHSM) - Administration Manual / Utimaco IS GmbH	2020-0040
	u.trust Anchor - csadm Manual / Utimaco IS GmbH	2021-0037
	CryptoServer – cxitool Manual / Utimaco IS GmbH	2018-0003
	CryptoServer - PKCS#11 p11tool2 - Reference Manual / Utimaco IS GmbH	2012-0004
	SecurityServer 4.50 - Elliptic Curves	2021-0051
[CSCXI]	CryptoServer Cryptographic Service Interface – Firmware Module CXI – Interface Specification / Utimaco IS GmbH	2008-0009
[FIPS140-3]	FIPS PUB 140-3, Security Requirements for Cryptographic Modules / National Institute of Standards and Technology (NIST)	
[FIPS186-4]	FIPS PUB 186-4, Digital Signature Standard/ National Institute of Standards and Technology (NIST)	
[PKCS11]	PKCS#11 v2.40 – Cryptographic Token Interface Standard, http://docs.oasis-open.org/pkcs11/	
[TR-03111]	TR-03111 Technical Guideline TR-03111 – Elliptic Curve Cryptography,; Version 1.11, April 2009 / Bundesamt für Sicherheit in der Informationstechnik (BSI)	
[NISTSP80090]	NIST Special Publication 800-90: Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised) / National Institute of Standards and Technology (NIST), March 2007	
[NISTSP80090A]	SP 800-90A Rev. 1: Recommendation for Random Number Generation Using Deterministic Random Bit Generators / National Institute of Standards and Technology (NIST), June 2015	
[NISTSP80090B]	SP 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation / National Institute of Standards and Technology (NIST), January 2018	
[ANSI-X9.62]	ANS X9.62-2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) / ANSI (American National Standards Institute)	

<i>Reference</i>	<i>Title/Company</i>	<i>Document Number</i>
[SEC2]	SEC2: Recommended Elliptic Curve Domain Parameters – Certicom Research – January 27, 2010, Version 2.0	
[SP80056A]	SP 800-56A Rev. 3: Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography/National Institute of Standards and Technology (NIST), April 2018	

6 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.