

CryptoServer

CryptoServer CSP and CryptoServer CNG Key Storage
Provider 1.x and 2.x

Manual for System Administrators



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	2.4.16
Product Version	6.0.0
Date	2024-10-23
Document No.	2008-0002
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	5
1.1	About this Manual	5
1.2	Document Conventions	5
2	CryptoServer CSP and CNG Key Storage Provider.....	7
2.1	Key Storage	7
2.1.1	Internal Key Storage	7
2.1.2	External Key Storage	8
2.2	Supported Algorithms	9
2.3	Key Access Restrictions	9
2.4	Windows Access Control for Keys	10
2.4.1	Access Rights for CSP Functions	12
2.4.2	Access Rights for CNG Functions.....	13
2.5	Key Management.....	14
3	Installation and Prerequisites	15
4	Configuration	16
4.1	CSP/CNG Configuration File	16
4.1.1	Location of the Configuration File	16
4.1.1.1	Configuration File Options	16
4.2	Authentication Mechanisms.....	20
4.3	Interactive Login.....	23
5	Key Management with the CNG Key Management Tool.....	26
5.1	Command Overview	26
5.2	Basic Commands	27
5.2.1	Help	27
5.2.2	EnumProvider	27
5.2.3	ProviderInfo	28
5.2.4	ListAlgos.....	29
5.3	Key Management Commands.....	30
5.3.1	ListKeys	30
5.3.2	KeyInfo	31
5.3.3	CreateKey	33
5.3.4	DeleteKey	35
5.3.5	ExportKey	36

5.3.6	ImportKey	38
5.4	Backup and Restore Commands	40
5.4.1	BackupKey	40
5.4.2	RestoreKey	41
5.4.3	RecryptExternalKeys	42
5.5	Migration Commands	43
5.5.1	CreateConfigFile	43
5.5.2	MigratePermissions	44
5.6	Managing CSP Containers with CNG tool	44
5.6.1	General Syntax Mapping.....	44
5.6.1.1	Creating a CSP Container for Key Exchange Purposes.....	45
5.6.1.2	Creating a CSP Container for Signature Purposes.....	45
5.6.2	Additional Notes	46
6	References	48

1 Introduction

Thank you for purchasing our CryptoServer security system (referred to below as CryptoServer). We hope you are satisfied with our product. Please do not hesitate to contact us if you have any questions or comments.

1.1 About this Manual

This document describes the CryptoServer CSP and CryptoServer CNG Key Storage Provider (referred to below as CSP/CNG Provider) for the hardware security module CryptoServer which is implemented and manufactured by Utimaco IS GmbH.

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 1: Document conventions

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



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



Here, you find additional notes or supplementary information.



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

2 CryptoServer CSP and CNG Key Storage Provider

The **CSP (Cryptographic Service Provider)** is a general purpose cryptography standard developed by Microsoft. At the top side it defines a cryptographic interface to be used by applications (CryptoAPI). On the bottom side it defines an interface to be used by manufacturers in order to integrate their cryptographic hardware. With this concept the application does not need to know about specific drivers to access cryptographic hardware directly.

The **CNG (Cryptography API Next Generation)** is the second generation cryptographic interface developed by Microsoft. It offers updated cryptographic algorithms and is intended as long term replacement of CSP.

Utimaco's **CryptoServer CSP** implements full RSA provider functionality. To use this interface, the firmware module CXI, `cx1.mtc`, must be loaded into the CryptoServer, as part of an appropriate firmware package. Additionally, Utimaco's CSP library, `cs2csp.dll`, and CNG library, `cs2cng.dll`, must be installed and registered on the host computer.

Utimaco's **CryptoServer CNG Key Storage Provider** implements a CNG key storage interface. To use this interface, the firmware module CXI, `cx1.mtc`, must be loaded into the CryptoServer, as part of an appropriate firmware package. Additionally, Utimaco's CNG library, `cs2cng.dll`, must be installed and registered on the host computer.

See the `<install dir>\Documentation\Product Details\CS_PD_SecurityServer_Supported_Platforms.pdf` file for details about the supported operating systems.

Please refer to Microsoft's MSDN web pages for a detailed specification of the CSP/CNG functionality [\[CNG\]](#) (p. 48).

2.1 Key Storage

Cryptographic keys that have been generated by or imported into the CryptoServer, can either be stored inside the CryptoServer (referred to as internal key storage) or on the host computer, outside the CryptoServer (referred to as external key storage). Both key storage types fulfill different requirements and are explained in the following two sections.

2.1.1 Internal Key Storage



The internal key storage offers the highest security level.

The internal key storage means that cryptographic keys are stored within the tamper-protected area of the CryptoServer. In case any physical or logical attack has been detected by the sensory of the CryptoServer, an alarm is triggered and all cryptographic keys are automatically deleted.

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

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

can be stored in the internal keystore of the CryptoServer. For CSP/CNG, always not only the private keys are stored but also the corresponding public keys are stored. A private key and a public key are stored in the same object. 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.

2.1.2 External Key Storage



The external key storage offers the highest level of flexibility.

The external key storage means that cryptographic keys are stored within a special directory on the disk drive of the host computer. Each key is stored as a separate key blob file (*.kb), which is encrypted and signed with the Master Backup Key of the CryptoServer, an AES 256-bit key referred to as MBK. As the MBK never leaves the CryptoServer, keys are protected nearly as secure as if stored internally. On demand a key is automatically loaded into the CryptoServer in order to perform a cryptographic operation. The number of keys which can be stored externally is only limited by the capacity of the disk drive.

As the CryptoServer remains completely stateless, fault tolerance is provided if multiple CryptoServer containing the same MBK (CryptoServer cluster) are used. In error cases (e.g., lost network connection to the primary server), the CSP/CNG Provider automatically switches to another CryptoServer. Except from a possible delay the application doesn't recognize this operation.

2.2 Supported Algorithms

The table below provides an overview of all cryptographic key and hash algorithms supported by the CSP/CNG provider.

Algorithm	Key/Hash length/Curves (bits)	Interface	
		CSP	CNG
DES	56, 112 and 168	✓	✓
AES	128, 192 and 256	✓	✓
RSA	512 – 16.384 (delta = 8 bit)	✓	✓
ECDSA	NIST-P256, NIST-P384, NIST-P521	✗	✓
ECDH	NIST-P256, NIST-P384, NIST-P521	✗	✓
MD5	128	✓	✓
SHA-1	160	✓	✓
RMD-160	160	✓	✓
SHA-256	256	✓	✓
SHA-384	384	✓	✓
SHA-512	512	✓	✓

Table 2: List of supported key algorithms

2.3 Key Access Restrictions

Access to keys can be restricted for certain users in order to use Utimaco's CNG Provider in a multi-client environment.

For this purpose, every key is unambiguously assigned to a key group, which has to be defined on key creation or import (an empty key group is only a special case, but no exception). This key group is defined by the `Group` entry in the configuration file (see [CSP/CNG Configuration File \(p. 16\)](#) for a detailed description of the configuration file).

The key access for a certain user is restricted on user creation by the user manager of the CryptoServer by adding the attribute `CXI_GROUP=<pattern>` to the user account. If this attribute has not been assigned to the user account, the user is not permitted to access/use any cryptographic keys. The pattern may contain wild cards to reflect graded access rights.

The following examples show the usage of the user attribute `CXI_GROUP =` for the definition of key access restrictions:

CXI_GROUP=	Description
Test-CA01	The user is allowed to access the key group 'Test-CA01'
Test-CA02	The user is allowed to access the key group 'Test-CA02'
Test-CA0?	The user is allowed to access every key group named 'Test-CA0x' (e.g., 'Test-CA01' and 'Test-CA02')

<i>CXI_GROUP=</i>	<i>Description</i>
Test-CA*	The user is allowed to access every key group beginning with 'Test-CA'
-CA	The user is allowed to access every key group containing the pattern '-CA' in the middle

Table 3: Examples for defining key access restrictions

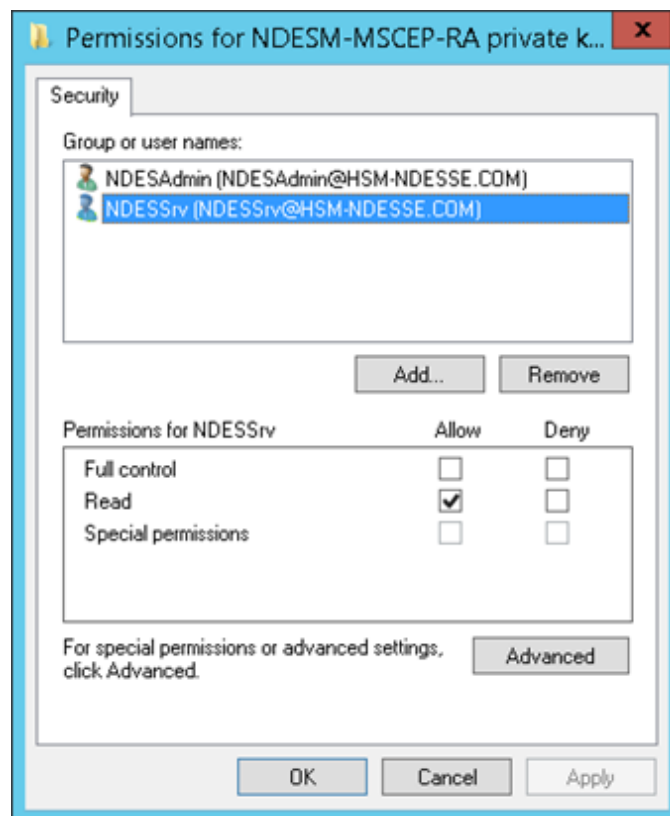
2.4 Windows Access Control for Keys

Windows provides the ability to restrict the access to CSP containers/CNG keys (referred to below also as containers/keys) by assigning permissions to specific users or groups of users. Utimaco's CSP/CNG Provider 2.x stores these permissions together with the key in the key storage and checks the corresponding, in the internal key database of the CryptoServer, `CXIKEY.db`, and checks the corresponding access rights when specific cryptographic operations shall be executed.



The CSP/CNG access rights check for cryptographic keys is only available starting with Utimaco's CSP/CNG Provider 2.0 provided on the SecurityServer/CryptoServer SDK product CD 4.10, and requires the CXI firmware module version 2.2.0.0 or later to be loaded on the CryptoServer.

Windows access rights for containers/keys are defined by security descriptors which can be set by using the `CryptSetProvParam()` function with the `PP_KEYSET_SEC_DESCR` parameter for CSP or by using the `NCryptSetProperty()` function with the `NCRYPT_SECURITY_DESCR_PROPERTY` property for CNG. For example, the Certificate Snap-in of the Microsoft Management Console calls these functions to provide a GUI for managing the private keys of installed certificates:



When a new CSP container is created or a new CNG key is created or imported, Utimaco's CSP/CNG provider uses the default owner and primary group of the current process as owner and group of the new container/key, and grants full control permissions for this owner to the container/key.

When a container/key is opened, its security descriptor is read by Utimaco's CSP/CNG provider. For performance reasons, the security descriptor is cached internally and not read again for every access check. The cache is only updated when the security descriptor is queried by using the `CryptSetProvParam()` function for CSP or the `NCryptSetProperty()` function for CNG, and is cleared when the container/key handle is released. Therefore, a process could still be able to access a container/key although the access rights have been changed in the meantime by another process.

Windows only distinguishes between read and full control access rights for containers/keys and so does Utimaco's CSP/CNG provider.

- A user or a group of users needs read permissions (`GENERIC_READ` in security descriptor terminology) to be able to read container/key properties and to use a key in cryptographic operations.
- A user or a group of users needs full control permissions (`GENERIC_ALL` in security descriptor terminology) to be able to change the container/key properties.

If the requested operation is not permitted, an error code (`NTE_PERM`) is returned.

2.4.1 Access Rights for CSP Functions

The following table lists the minimal access rights which are necessary to execute specific CSP functions.



Functions which are not listed in that table do not require any specific access rights.

<i>CSP Function</i>	<i>Required Access Right</i>
CryptAcquireContext	<ul style="list-style-type: none"> ▪ Full control if <code>CRYPT_DELETEKEYSET</code> flag is set ▪ Read unless the default container is to be opened or the <code>CRYPT_DELETEKEYSET</code> or <code>CRYPT_NEWKEYSET</code> flag is set
CryptDecrypt	Read if it is a persistent key (see <code>CryptGenKey</code>)
CryptEncrypt	Read if it is a persistent key (see <code>CryptGenKey</code>)
CryptExportKey	Read if it is a persistent key (see <code>CryptGenKey</code>). This holds for both the key to be exported and the export key. Note that for the export key the access rights of the key are checked directly and not the rights of the container, i.e. changing the container access rights after export key has been opened has no effect.
CryptGenKey	Full control if a <code>AT_KEYEXCHANGE</code> or <code>AT_SIGNATURE</code> key is to be created and <code>CRYPT_VOLATILE</code> flag is not set, i.e. for persistent keys only
CryptGetKeyParam	Read if it is a persistent key (see <code>CryptGenKey</code>)
CryptGetProvParam	Read if <code>PP_KEYSET_SEC_DESCR</code> is queried
CryptGetUserKey	Read
CryptHashSessionKey	Read if it is a persistent key (see <code>CryptGenKey</code>)
CryptImportKey	Full control if a Utimaco Backup BLOB is to be imported Full control for import container if a <code>AT_KEYEXCHANGE</code> or <code>AT_SIGNATURE</code> key is to be imported and the <code>CRYPT_VOLATILE</code> flag is not set, i.e. for persistent keys only; read for import key

CSP Function	Required Access Right
CryptSetKeyParam	Full control if it is a persistent key (see <code>CryptGenKey</code>)
CryptSetProvParam	Full control if <code>PP_KEYSET_SEC_DESCR</code> is to be set
CryptSignHash	Read
CryptVerifySignature	Read if it is a persistent key (see <code>CryptGenKey</code>)

Table 4: Minimal access rights to execute CSP functions

2.4.2 Access Rights for CNG Functions

The following table lists the minimal access rights which are necessary to execute specific CNG functions.



Functions which are not listed in that table do not require any specific access rights.



In case key attributes are displayed as ' ??? ' when executing a `cngtool` command, this means that the user executing the `cngtool ListKeys` command does not have the right Windows permissions to access the key.

CNG Function	Required Access Rights
NCryptCreatePersistedKey	Full control if the key exists and the <code>NCRYPT_OVERWRITE_KEY_FLAG</code> is set. Check is done on <code>NCryptFinalizeKey</code> .
NCryptDecrypt	Read
NCryptDeleteKey	Full control
NCryptEncrypt	Read
NCryptExportKey	Read
NCryptFinalizeKey	See <code>NCryptCreatePersistedKey</code> and <code>NCryptImportKey</code>
NCryptGetProperty	Read if used with a key handle

CNG Function	Required Access Rights
NCryptImportKey	Full control for a key to be imported if the key exists and the <code>NCRYPT_OVERWRITE_KEY_FLAG</code> is set; read for import key Check is done on <code>NCryptFinalizeKey</code> if <code>NCRYPT_DO_NOT_FINALIZE_FLAG</code> is set.
NCryptOpenKey	Read
NCryptSecretAgreement	Read for both private and public key used in a secret agreement
NCryptSetProperty	Full control if used with a key handle
NCryptSignHash	Read
NCryptVerifySignature	Read

Table 5: Minimal access rights to execute CNG functions

2.5 Key Management

Utimaco also provides an appropriate tool for managing cryptographic keys. This tool can be found on the SecurityServer and the CryptoServer SDK product CDs in the directory `\Software\Windows\<x86-xx>\Administration\`.

- CNG Tool is the command-line tool to be used for managing the cryptographic keys used by Utimaco's CNG Key Storage Provider.

This tool is installed by the CryptoServer host software installer when selecting the installation of the **CSP/CNG – Cryptographic Service Provider for Microsoft Windows**. By default, the installation directory for SecurityServer/CryptoServer SDK, which is automatically added to the PATH environment variable is:

```
C:\Program Files\Utimaco\SecurityServer\Administration\
```

To start using the CNG tool, just open a command prompt and type `cngtool <command>`. For a list of available commands, see [Key Management with the CNG Key Management Tool](#) (p. 26).

3 Installation and Prerequisites

For the proper operation of the CryptoServer CSP and CryptoServer CNG Key Storage Provider the following requirements must be met:

- One or more CryptoServer devices are available – either locally in form of a PCIe card mounted in a host computer, or remotely in form of a CryptoServer LAN network appliance.
- The firmware module CXI and all basic system firmware modules are loaded and running on the CryptoServer device(s). This requires that the appropriate firmware module package for your CryptoServer model range, `SecurityServer-<type>-<version>.mpkg`, is loaded into the CryptoServer. Here `<type>` denotes the model range of the CryptoServer (CSe-Series or Se2-Series) and `<version>` stands for the version number of the firmware package. To load the corresponding firmware package into the CryptoServer, see sections *Commands for Firmware Management* and *Commands for Firmware Packaging* in the *CryptoServer - csadm Manual* for help.
- The CryptoServer host software is installed.

The CryptoServer installer software, `SecurityServer-X.XX.X.X.msi` (for CryptoServer 4.40.0 and later) or `CryptoServerSetup-X.XX.X.X.exe` (for CryptoServer versions earlier than 4.40.0), which you find on the delivered product CD, automatically copies the necessary files and registers the provider. The CSP/CNG Interface must be selected when running the installer.

- The CSP/CNG provider must be configured as described in section one of [CSP/CNG Configuration File \(p. 16\)](#).

4 Configuration

As of SecurityServer/CryptoServer SDK 4.10, the CSP/CNG Provider 2.x is configured by the CSP/CNG configuration file, which is described in [CSP/CNG Configuration File \(p. 16\)](#). This configuration file replaces the CSP/CNG Control Applet, which was provided with the SecurityServer/CryptoServer SDK product CD 4.01 and earlier to configure the CSP/CNG Provider 1.x.

For details about the migration from CSP/CNG Provider 1.x to CSP/CNG Provider 2.x, see earlier versions of this documentation.

4.1 CSP/CNG Configuration File

The configuration file replaces the CSP/CNG Control Panel Applet starting with CSP/CNG Provider 2.x provided as of SecurityServer/CryptoServer SDK 4.10.

4.1.1 Location of the Configuration File

The environment variable `CS_CNG_CFG` contains the path and name of the configuration file. It is set by default during the host software installation to

```
C:\ProgramData\Utlimaco\CNG\cs_cng.cfg
```



The `ProgramData` directory is hidden by default. To access the CNG configuration file, enter the path mentioned above in the address bar of the Windows Explorer.


Alternatively, it can be set/changed manually, for example:

```
C:\>set CS_CNG_CFG=<customized location>\cs_cng.cfg
```

There is no default location for the configuration file. If the environment variable is not set or the defined configuration file cannot be found or read, an error message is written to the Windows Event Log.

4.1.1.1 Configuration File Options

The following table gives an overview of all parameters that can be configured in the CNG configuration file.

Parameter	Description
Logging	<p>Specifies the log level. The following values are valid:</p> <ul style="list-style-type: none"> 0: No logging output 1: Log errors only 2: Log errors and warnings 3: Log errors, warnings and additional information of the CNG provider 4: Log errors, warnings, additional information and trace output like function calls and parameters of the CNG provider <hr/> <p> Do not use log level higher than 2 in production environments.</p> <hr/>
Logpath	<p>Specifies the path where the log file shall be created. The directory must be created by the user and be given appropriate rights. The filename <code>cs2cng.log</code> is appended automatically by the CSP/CNG Provider.</p>
Logsize	<p>Defines the maximum size of the log file. If the maximum is reached, the old log file will be renamed to <code>cs2cng.log.01</code> and a new log file <code>cs2cng.log</code> is created. If a file <code>cs2cng.log.01</code> already exists, this file will be renamed to <code>cs2cng.log.02</code> and so on. Nine backup log files are kept at maximum.</p> <p>The file size can be defined as a value in bytes or as a formatted text. Valid text formats are: <code>kb</code>, <code>mb</code>, <code>gb</code></p> <p>Examples:</p> <p><code>Logsize = 1000</code> means that the log size is 1000 bytes.</p> <p><code>Logsize = 1000kb</code> means that the log size is 1000 kilobytes.</p>
KeysExternal	<p>Specifies the type of the key storage: internal or external. This setting is of type Boolean.</p> <p>If <code>KeysExternal = true</code>, the keys are only read and written from/to an external key storage, i.e., a key database outside the CryptoServer and protected with the Master Backup Key of the CryptoServer.</p> <p>If <code>KeysExternal = false</code>, the keys are only read/written from/to the internal key database of the CryptoServer.</p>
KeyStore	<p>Specifies the path to the external key storage (default <code>C:\ProgramData\Utimaco\CNG\keys</code>). This parameter shall be set if <code>KeysExternal = true</code>.</p> <p>The directory must be created by the user and be given appropriate rights. The filename of the key storage is appended automatically by the CNG provider.</p>

<i>Parameter</i>	<i>Description</i>
ExportPolicy	<p>Defines which export properties cannot be set by a CNG user.</p> <ul style="list-style-type: none"> ▪ 0 <ul style="list-style-type: none"> • For CNG, the key export policy is defined by the <code>NCRYPT_ALLOW_EXPORT_FLAG</code> and the <code>NCRYPT_ALLOW_PLAINTEXT_EXPORT_FLAG</code> in a <code>NCryptSetProperty()</code> function call • CSP keys are exportable as plaintext if created with the <code>CRYPT_EXPORTABLE</code> flag. ▪ 1 <ul style="list-style-type: none"> • For CNG, this parameter denies the plaintext export of newly generated keys by overruling the <code>NCRYPT_ALLOW_PLAINTEXT_EXPORT_FLAG</code> in a <code>NCryptSetProperty()</code> function call. • CSP keys created with the <code>CRYPT_EXPORTABLE</code> flag can be exported when wrapped with another key. ▪ 2 <ul style="list-style-type: none"> • For CNG, this parameter denies both encrypted and plaintext export of newly generated keys by overruling both the <code>NCRYPT_ALLOW_EXPORT_FLAG</code> and the <code>NCRYPT_ALLOW_PLAINTEXT_EXPORT_FLAG</code> in a <code>NCryptSetProperty()</code> function call. • For CSP keys the <code>CRYPT_EXPORTABLE</code> flag will be overruled and key export is denied completely.
Group	<p>Defines the key group, which is assigned to new keys and to which keys shall belong in order to be accessible to the CryptoServer CSP/CNG Provider.</p>
ConnectionTimeout	<p>Specifies the maximum time in milliseconds to wait before the connection establishment is aborted if the device is not responding. Default value is 3000 ms. In practice, the timeout can reach approximately $2 \cdot n \cdot T$. Legend: - n: Number of HSMs specified by the <code>Device</code> parameter - T: The timeout value specified by the <code>ConnectionTimeout</code> parameter.</p>
CommandTimeout	<p>Specifies the maximum time in milliseconds to wait for the answer from CryptoServer after sending a command. Default value is 60000 ms. In practice, the timeout can reach approximately $2 \cdot n \cdot T$. Legend: - n: Number of HSMs specified by the <code>Device</code> parameter - T: The timeout value specified by the <code>CommandTimeout</code> parameter.</p>
KeepAlive	<p>Keep sessions alive and prevent them from expiring after 15 minutes idle time. This setting is of type Boolean. Default value is <code>true</code>.</p>

Parameter	Description
Device	<p>Specifies the device address of the CryptoServer device to connect to the CSP/CNG Provider. This parameter can only specify a single device address per statement. There might be multiple Device= statements in the CSP/CNG Provider configuration file. The first Device= statement defines the default device. A connection to the next device(s) is automatically requested, if the previous one(s) does not respond. Valid device specifiers:</p> <ul style="list-style-type: none"> Device=PCI:0 for a CryptoServer PCIe card Device=3001@127.0.0.1 for the CryptoServer Simulator Device=<IPv4 or IPv6 address> for a CryptoServer LAN <p>For details, see the next table.</p>
Login	<p>Specifies the encrypted or plaintext authentication credentials for the CryptoServer CNG user who is the only user permitted to generate/access cryptographic keys. At the first installation of the CSP/CNG Provider 2.x, this user shall be manually created with one of the CryptoServer's administration tools – csadm or CAT – as a user with the Cryptographic User profile with the permissions 00000002, the attribute CXI_GROUP=<Group> and one of the authentication mechanisms. See section <i>AddUser</i> in the <i>CryptoServer - csadm Manual</i> for details. For details about the authentication mechanisms, see also Authentication Mechanisms (p. 20).</p> <p>An example with csadm:</p> <pre>csadm Dev=PCI:0 LogonSign=UserAdmin,:cs2:cio:USB0 AddUser=CNGUser,002{CXI_GROUP=CNG},hmacpwd,ask</pre> <p>If the Login entry is not present in the configuration file, the CSP/CNG provider displays an interactive login dialog. If the error message <code>Could not connect to cs2cngsrv, please make sure the process is running to enable user interactions</code> is shown, make sure that the <code>cs2cngsrv.exe</code> file is started. See Interactive Login (p. 23) for further details.</p> <p>If a user with HMAC password authentication is created, this user cannot perform commands he/she needs an authentication for (except for the <code>csadm ChangeUser</code> command). To enable the user to perform commands he/she needs an authentication for, the user must change his/her credentials. To do so, this user must perform the <code>csadm ChangeUser</code> command. It is important that the changes are performed by the user himself/herself and not, for example, by an administrator.</p>

Table 6: CNG Configuration File Parameters

The following table describes the `Device` parameter in more detail.

Device address	Description
<code>/dev/cs2.n</code> where <code>n = {0, 1, 2, ..., 7}</code>	<p>Local CryptoServer No. <code>n+1</code> on a UNIX system.</p> <p>The maximum number of eight CryptoServer PCIe cards can be changed in the source of the Linux driver.</p>

Device address	Description
PCI:n where n = {0, 1, 2, ..., 31}	Local CryptoServer No. n+1 on a Windows system
TCP:288@194.168.4.107	IP address and port number of a CryptoServer LAN In csadm commands, always use IP addresses without leading zeros although they are shown in the CryptoServer LAN display, e.g., 194.168.004.107.
TCP:194.168.4.107	IP address of a CryptoServer LAN (default: port=288) In csadm commands, always use IP addresses without leading zeros although they are shown in the CryptoServer LAN display, e.g., 194.168.004.107.
194.168.4.107	IP address of a CryptoServer LAN (default: protocol=TCP, port=288) In csadm commands, always use IP addresses without leading zeros although they are shown in the CryptoServer LAN display, e.g., 194.168.004.107.
TCP:288@cslan01	Host name and port number of a CryptoServer LAN (using DNS request to resolve host name)
TCP:cslan01	Host name of a CryptoServer LAN (using DNS request to resolve host name, default: port=288)
cslan01	Host name of a CryptoServer LAN (using DNS request to resolve host name, default: protocol=TCP, port=288)
TCP:3001@127.0.0.1 or TCP:3001@localhost	Local CryptoServer simulator for Windows/Linux (SDK)
3001@127.0.0.1 or 3001@localhost	Local CryptoServer simulator for Windows/Linux (SDK) with the default protocol TCP

Table 7: Device Parameter

4.2 Authentication Mechanisms

If you want to use an encrypted password, use the `cngtool EncryptAuthToken` command to create the hexadecimal representation of this encrypted password in the output of this command. Use this encrypted password to replace `<encrypted password>` in the following examples.

The following list shows the various authentication mechanisms supported by the CryptoServer and how they are specified in the Login entry of the CSP/CNG configuration file. See the section *Authentication Mechanisms* in the *CryptoServer - Administration Manual* for more detailed information about authentication mechanisms.

- HMAC password authentication

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



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

If a user with HMAC password authentication is created, this user cannot perform commands he/she needs an authentication for (except for the `csadm ChangeUser` command). The `csadm ListUser` command shows this user with the `I[1]` attribute value. To enable the user to perform commands he/she needs an authentication for, the user must change his/her credentials. To do so, this user must perform the `csadm ChangeUser` command. If the user is a PKCS#11 Security Officer (SO) or a PKCS#11 normal user, he/she may perform the `p11tool2 SetPIN` command as an alternative. It is important that the changes are performed by the user himself/herself and not, for example, by an administrator. The `csadm ListUser` command then shows this user with the `I[0]` attribute value.

- Plaintext password

```
Login = <user name>, <plaintext password>
```

```
Login = <user name>,HMACPwd=<plaintext password>
```

- Encrypted password

```
Login = <user name>,HMACPwdEnc= <encrypted password>
```

- RSA signature authentication with a keyfile

The password entry is optional. If the keyfile is encrypted/password-protected, the password must be given with a preceding #.

- Plaintext password

```
Login = <user name>,RSASign=<path file name>#<plaintext password>
```

- Encrypted password

```
Login = <user name>,RSASignEnc=<path file name>#<encrypted password>
```

- RSA signature authentication with a smartcard; the PIN pad for reading the content of the smartcard is connected to a host computer.

Optionally, the PIN can be given with a preceding # for automatic login. If the PIN is not present, the user must enter it via the keyboard of the PIN pad. See the sections *Key Specifiers* the *CryptoServer - csadm Manual* and *Using a Local PIN Pad for a Remote CryptoServer* in the *CryptoServer - Administration Manual* for details about key specifiers.

- Plaintext PIN

```
Login = <user name>,RSASign=<key specifier>#<plaintext PIN>
```

- Encrypted PIN

```
Login = <user name>,RSASignEnc=<key specifier>#<encrypted PIN>
```

- RSA smartcard authentication (with a smartcard; the PIN pad for reading the content of the smartcard is directly connected to the CryptoServer PCIe card)

Optionally, the PIN can be given with a preceding # for automatic login. If the PIN is not present, the user must enter it via the keyboard of the PIN pad. See the section *Key Specifiers* in the *CryptoServer - csadm Manual* for details about key specifiers. The section *Using a Local PIN Pad for a Remote CryptoServer* in the *CryptoServer - csadm Manual* does not apply here because a directly connected PIN pad is a must.

- Plaintext PIN

```
Login = <user name>,RSASC=<key specifier>#<plaintext PIN> Encrypted PIN
```

```
Login = <user name>,RSASCEnc=<key specifier>#<encrypted PIN>
```

- ECDSA signature authentication with a keyfile

The password entry is optional. If the keyfile is encrypted, the password must be given with a preceding #.

- Plaintext password

```
Login = <user name>,ECDSA=<path file name>#<plaintext password>
```

- Encrypted password

```
Login = <user name>,ECDSAEnc=<path file name>#<encrypted password>
```

- ECDSA signature authentication with a smartcard; the PIN pad for reading the content of the smartcard is connected to the host computer:

Optionally, the PIN can be given with a preceding # for automatic login. If the PIN is not present, the user must enter it via the keyboard of the PIN pad. See the sections *Key Specifiers* and *Using a Local PIN Pad for a Remote CryptoServer* in the *CryptoServer - Administration Manual* for details about key specifiers.

- Plaintext PIN

```
Login = <user name>,ECDSA=<key specifier>#<plaintext PIN>
```

- Encrypted PIN

```
Login = <user name>,ECDSAEnc=<key specifier>#<encrypted PIN>
```

Examples:

Only one of the following examples can be applied at a time. The other entries are comments due to the leading #.

```
Login =
HSM6037,HMACPwdEnc=01fcd3c62435490ad966199ad8d1e8339f43376e33a307d804672019
e80d22784f52c53bc951bc5517353648aab1724a
#Login = cnguser,utimaco
#Login = cnguser,HMACPwd=utimaco
#Login = cnguser,HMACpwdEnc=B497126F263662434626C5276F920897
#Login = cnguser2,RSASign=D:\rsa_key_enc.key#utimaco123
#Login = cnguser2,RSASignEnc=D:
\rsa_key_enc.key#7EFA9053214093D00BBBCEEA2069644C
```

4.3 Interactive Login

If the Login entry is not present in the configuration file and if the currently logged in Windows user has the login ownership, the CSP/CNG provider displays an interactive login dialog to log in to the CryptoServer.

Only one Windows user can have the login ownership at a time although several Windows users can be logged in to this computer. A Windows user must request the login ownership explicitly as described below. The login ownership does not automatically move from one user to another. The interactive login dialog is always shown to the Windows user who has the login ownership even if the CNG/CSP request that initiates this interactive login dialog is performed by another Windows user.

If a Windows user A has the login ownership and opened the interactive login dialog, another Windows user B can request the login ownership but this request only is granted when Windows user A closes the interactive login dialog.

If the error message `Could not connect to cs2cngrsv, please make sure the process is running to enable user interactions` is shown, make sure that the `cs2cngrsv.exe` file is started.

Perform the following steps to provide the login ownership.

1. Verify whether there is the Utimaco Interactive CNG login tray icon in the lower right corner of the Windows screen.

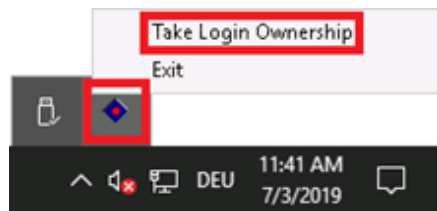


Figure 1 : Utimaco Interactive CNG login tray icon

If this tray icon has a red center point, you do not have the CNG login ownership, that means, that you cannot log in to the CryptoServer interactively. If this tray icon has a green center point, you have the CNG login ownership and you can log in interactively. If this tray icon is not shown, make sure that the `cs2cngrsv.exe` file is started.

2. Click this tray icon with the right mouse button and select **Take Login Ownership**. The color of the center point changes to green and a notification message is shown that the ownership has been acquired.



Figure 2 : Login Ownership

If you select **Exit**, `cs2cngrsv` stops.

3. Now you can perform a CSP/CNG request and log in interactively.

[illegible]

Figure 3 : Interactive login dialog

5 Key Management with the CNG Key Management Tool

This chapter is a reference to the commands of the CNG command-line tool and their use.



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

For example, a correct entry for the `cngtool CreateKey` command in a Microsoft PowerShell:

```
cngtool Export=allow Name=DemoRSA CreateKey="RSA,2048"
```

5.1 Command Overview

The following table gives a short overview of the CNG tool commands.

Command	Description
Basic Commands	
Help	Show a list of all available commands if called without any parameter or specific help if a command name
EnumProvider	Display a list of all available Cryptographic Key Storage Provider
ProviderInfo	Retrieve information about the providers' name and version, implementation details of the key storage pro
ListAlgos	Retrieve a list of all cryptographic algorithms supported by a specific Key Storage Provider
Key Management Commands	
ListKeys	Retrieve a list of all cryptographic keys stored on the Key Storage Provider
KeyInfo	Retrieve detailed information about a specific key stored on the Key Storage Provider.
CreateKey	Generate a specified key of type RSA or ECDSA in the Key Storage Provider database of the specific Key S
DeleteKey	Delete a specified key from the Key Storage Provider.
ExportKey	Export a Microsoft key blob into a file
ImportKey	Import a Microsoft key blob from a file
Backup/Restore Commands	

Command	Description
Basic Commands	
BackupKey	Create a backup copy of a cryptographic key internally stored in the database of the CryptoServer CNG Key
RestoreKey	Restore a cryptographic key from a backup copy into the internal key database of the CryptoServer CNG Ke

Table 8: Overview of CNG tool commands

5.2 Basic Commands

5.2.1 Help

If called without any parameter, this command shows a list of all available cngtool commands (except for a few rarely used specific ones described in this document). If a command name is given as a parameter, specific help will be provided.

Syntax	<pre>cngtool Help cngtool Help=<command></pre>
---------------	--

Parameter	Description
<command>	Specific cngtool command

Example	<pre>cngtool Help=ListKeys</pre>
----------------	----------------------------------

Output	<p>Upon successful execution of the command, the description and parameters of the specified command are returned.</p> <p>List keys stored on Key Storage Provider</p> <p>syntax:</p> <pre>cngtool [Provider=<provider_name>] [Machine] [Name=<key_name>] ListKeys</pre>
---------------	--

5.2.2 EnumProvider

This command displays a list of all available Cryptographic Key Storage Provider.

Syntax	<pre>cngtool EnumProvider</pre>
---------------	---------------------------------

Example	<code>cngtool EnumProvider</code>
----------------	-----------------------------------

Output	Upon successful execution of the command, a list of all available Cryptographic Key Storage Providers is returned. Microsoft Primitive Provider Microsoft Smart Card Key Storage Provider Microsoft Software Key Storage Provider Microsoft SSL Protocol Provider Utimaco CryptoServer Key Storage Provider
---------------	--

5.2.3 ProviderInfo

With this command you can retrieve information about the providers' name and version, implementation details of the key storage provider and the maximum length, in characters, of the name of a persistent key.

Syntax	<code>cngtool [Provider=<provider_name>] ProviderInfo</code>
---------------	--

Parameter	Description
<provider_name>	Default setting is Utimaco CryptoServer Key Storage Provider. If the parameter <code>Provider=</code> is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed. The set of valid provider names is retrieved by executing <code>cngtool EnumProvider</code> .

Example	<code>cngtool ProviderInfo</code>
----------------	-----------------------------------

Output	Upon successful execution of the command, information on the provider is returned.	
	<pre> ----- Provider : Utimaco CryptoServer Key Storage Provider Device : PCI:0 Group : DE-ANH1 Mode : External Key Storage ----- Name : Utimaco CryptoServer Key Storage Provider Version : 0x01060600 Impl.-Type : 0x00000011 MaxNameLength : 0x00000104 Device : PCI:0 Group : DE-ANH1 Mode : External Key Storage </pre>	

5.2.4 ListAlgos

This command displays a list of all cryptographic key algorithms supported by the given key storage provider. Additionally, information about the class of algorithms the given algorithm belongs to, and the operational class of the algorithm is provided.

Syntax	<code>cngtool [Provider=<provider_name>] ListAlgos</code>
---------------	---

Parameter	Description
<provider_name>	<p>Default setting is Utimaco CryptoServer Key Storage Provider.</p> <p>If the parameter <code>Provider=</code> is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed.</p> <p>The set of valid provider names is retrieved by executing <code>cngtool EnumProvider</code>.</p>

Example	<code>cngtool Provider="Utimaco CryptoServer Key Storage Provider" ListAlgos</code>
----------------	---

Output	Upon successful execution of the command, information on the algorithms supported by the provider is returned. Example output:
	<pre> ----- Provider : Utimaco CryptoServer Key Storage Provider Device : 3001@127.0.0.1 Group : DE-ANH1 Mode : External Key Storage ----- Name : RSA AlgClass : NCrypt_Asymmetric_Encryption_Interface AlgOperations : NCrypt_Asymmetric_Encryption_Operation : NCrypt_Signature_Operation Flags : 0x00000000 Name : ECDH_P256 AlgClass : NCrypt_Secret_Agreement_Interface AlgOperations : NCrypt_Secret_Agreement_Operation : NCrypt_Signature_Operation Flags : 0x00000000 Name : ECDH_P384 AlgClass : NCrypt_Secret_Agreement_Interface AlgOperations : NCrypt_Secret_Agreement_Operation : NCrypt_Signature_Operation Flags : 0x00000000 ... </pre>

5.3 Key Management Commands

5.3.1 ListKeys

This command lists detailed information about the keys stored on the given Key Storage Provider.

Syntax	<code>cngtool [Provider=<provider_name>] [Machine] [Name=<pattern>] ListKeys</code>
---------------	---

Parameter	Description
<provider_name>	Default setting is Utimaco CryptoServer Key Storage Provider. If the parameter <code>Provider=</code> is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed. The set of valid provider names is retrieved by executing <code>cngtool EnumProvider</code> .
Machine	This parameter defines a machine-specific keyset instead of a user-specific one. It is ignored by Utimaco's CSP/CNG Provider.

Parameter	Description
Name=<pattern>	Specifies the search pattern for the key, which can be the exact key name or a name ending with the character '*'. In the second case, all keys starting with the given name are displayed.

If none of the parameters is specified, all keys stored on Utimaco's CSP/CNG Provider are displayed.

Example	<code>cngtool Name=EXMP* ListKeys</code>
----------------	--

Output	Upon successful execution of the command, a list of keys is returned.			
	Example output:			
	Index	AlgId	Size	Group
	Name		Spec	

	1	RSA	512	DE-HSM1
	EXMPrsa		0	
	2	RSA	528	DE-HSM1
	EXMP528		0	
3	ECDSA_P384	384	DE-HSM1	
EXMPecdsa		3		



In case key attributes are displayed as ' ??? ', this means that the user executing the `cngtool ListKeys` command does not have the right Windows permissions to access the key. For CNG keys, a security descriptor is assigned to the keys when they are created and it is checked against the user that requests a key later on. For details, see [Windows Access Control for Keys \(p. 10\)](#).

5.3.2 KeyInfo

With this command detailed information about a specific key stored on the given Key Storage Provider can be retrieved.

Syntax	<code>cngtool [Provider=<provider_name>] [Machine] Name=<key_name> [Spec=<key_specifier>] KeyInfo</code>
---------------	--

Parameter	Description
<provider_name>	Default setting is Utimaco CryptoServer Key Storage Provider, optional. If the parameter <code>Provider=</code> is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed. The set of valid provider names is retrieved by executing <code>cngtool EnumProvider</code> .
Machine	This parameter defines a machine-specific keyset instead of a user-specific one, optional. It is ignored by Utimaco's CSP/CNG Provider.
Name=<key_name>	Specifies the name of the key
Spec=<key_specifier>	The key specifier as defined for the key on key generation, optional. Default value is 0.

Example	<code>cngtool Name=EXMPrsa KeyInfo</code>
----------------	---

Output	<p>Upon successful execution of the command, information on the specified key is returned. Example output:</p> <pre> ----- Provider : Utimaco CryptoServer Key Storage Provider Device : 3001@127.0.0.1 Group : DE-ANH1 Mode : External Key Storage ----- 1.: Algid : RSA Name : EXMPrsa Spec : 00000000 Flags : 00000000 Size : 512 Uname : 139B8621C7A052F4A8D539FFF2E4C6F4 AlgoGroup : RSA Export : 0x00000001 [+ ALLOW] Usage : 0x000002bb [+ DECRYPT + SIGN] Type : 0x00000000 [User] Block Length : 0x00000040 </pre>
---------------	--

The fields of the key information output have the following meaning:

Parameter	Description
Provider	Name of the currently used Key Storage Provider
Device	Address of the device to which the Key Storage Provider is connected
Group	Name of the group to which the key resp. the user creating the key belong

<i>Parameter</i>	<i>Description</i>
Mode	Type of the used key storage: external or internal (see also Key Storage (p. 7))
Algid	Algorithm identifier (for example, RSA, ECDSA_P256, ECDSA_P384, ECDSA_P521)
Name	Key name defined on key creation
Spec	Key specifier, in hexadecimal format, defined on key creation
Flags	Special key flags. No flags are defined for Utimaco's CSP/CNG Provider. For other providers, the Machine flag will be shown as 00000020 .
Size	Key size in bits
Uname	Unique key identifier. It is used as filename if the key is stored in an External Key Storage
AlgoGroup	Defines the group of algorithms the key belongs to
Export	Export policy as defined in the CNG configuration file or on key creation (in hexadecimal format); see Configuration File Options (p. 16)
Usage	Key usage policy as defined on key creation (in hexadecimal format). Describes for which cryptographic functions the key can be used: decryption/encryption, signature generation, secret agreement or all of them.
Type	Hexadecimal. Specifies the key type. For Utimaco CSP/CNG provider, the type is always 0x00000000 , which denotes a User key. For other providers, 0x00000001 denotes a Machine key.
Block Length	Block length of the key in hexadecimal format

Table 9: KeyInfo output parameters

5.3.3 CreateKey

This command generates a new key on the specified Key Storage Provider.

When using the `cngtool CreateKey` command to create a cryptographic key in an external database, consider that this external database is encrypted with the current Master Backup Key (MBK) in MBK slot 3. If you then change the MBK in this MBK slot (`csadm MBKImportKey` command) and you then try to show the available cryptographic keys (`cngtool ListKeys` command), the following error message is shown.

```
Error B0680080
```

```
CryptoServer module CXI
```

```
key blob encrypted with different MBK
```

For details about MBKs, see Section *Commands for Managing the Master Backup Keys* in the *CryptoServer - csadm Manual*.

Syntax	<code>cngtool [Provider=<provider_name>] [Machine] [Export=<export>] [Usage=<usage>] Name=<key_name> [Spec=<key_specifier>] CreateKey=<key_type>,<bit_size></code>
---------------	--

Parameter	Description
<provider_name>	Default setting is Utimaco CryptoServer Key Storage Provider, optional. If the parameter <code>Provider=</code> is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed. The set of valid provider names is retrieved by executing <code>cngtool EnumProvider</code> .
Machine	This parameter defines a machine-specific keyset instead of a user-specific one, optional. It is ignored by Utimaco's CSP/CNG Provider.
Export=<export>	This optional parameter defines whether the key can be exported or not. If the export policy in the CNG configuration forbids a certain export type, the next possible lower type is chosen. For example, if <code>allow_plain</code> is requested but the configuration file denies plaintext export and allows encrypted export, <code>allow</code> is selected for the new key. Possible values: <ul style="list-style-type: none"> <code>deny</code> - denies the export of the key in both encrypted form and in plaintext format <code>allow</code> - the key can be exported in encrypted form <code>allow_plain</code> - the key can be exported in encrypted or plaintext format If the <code>Export</code> property is omitted, <code>allow</code> is set by default.
Usage=<usage>	This optional parameter describes for which cryptographic functions the key can be used. Possible values: <ul style="list-style-type: none"> <code>decrypt</code> - decryption/encryption <code>sign</code> - signature generation <code>agree</code> - secret agreement <code>all</code> - decryption/encryption, signature generation, secret agreement Multiple values can be given as comma separated list. Default setting for an RSA key (if <code>Usage=<usage></code> is omitted) is <code>decrypt, sign</code> . Default setting for an ECDSA key is <code>all</code> .
Name=<key_name>	Specifies the name of the key
Spec=<key_specifier>	The key specifier as defined for the key on key generation, optional. Default value is 0.

Parameter	Description
<key_type>	RSA or ECDSA
<bit_size>	Key size in bit For RSA – 512 to 16.384 (delta = 8 bit), for ECDSA – 256, 384, 521

Example	<code>cngtool Export=allow Name=DEMOecdsa CreateKey=ECDSA,521</code>
----------------	--

Output	<p>Upon successful execution of the command, a new key is created. Example Output:</p> <pre> ----- Provider : Utimaco CryptoServer Key Storage Provider Device : 3001@127.0.0.1 Group : DE-ANH1 Mode : External Key Storage ----- </pre>
---------------	--



Certain types of shell processes treat certain characters (for example, commas, colons, semi-colons) differently. If the execution of the `cngtool CreateKey` command fails with an error message from the shell about missing parameter (Algo,Size) or illegal parameter format, quoting parameter values may be necessary.

For example, a correct command entry in the Microsoft PowerShell:

```
cngtool Export=allow Name=DemoRSA CreateKey="RSA,2048"
```

5.3.4 DeleteKey

This command removes a specified key from the key storage of the specified Key Storage Provider.

Syntax	<code>cngtool [Provider=<provider_name>] [Machine] Name=<pattern> [Spec=<key_specifier>] DeleteKey</code>
---------------	---

Parameter	Description
<provider_name>	<p>Default setting is Utimaco CryptoServer Key Storage Provider, optional.</p> <p>If the parameter <code>Provider=</code> is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed.</p> <p>The set of valid provider names is retrieved by executing <code>cngtool EnumProvider</code>.</p>

Parameter	Description
Machine	This parameter defines a machine-specific keyset instead of a user-specific one, optional. It is ignored by Utimaco's CSP/CNG Provider.
Name=<pattern>	Specifies the search pattern for the key(s) to be removed, which can be the exact key name or a string ending with the character '*'. In the second case, all keys starting with the given string are deleted.
Spec=<key_specifier>	Optional. Default value is 0.
<bit_size>	Key size in bit For RSA – 512 to 16.384 (delta = 8 bit), for ECDSA – 256, 384, 521

Example	cngtool Name=DEMOecdsa DeleteKey
----------------	----------------------------------

Output	<p>On successful execution of the command, a success message is returned.</p> <pre> Provider : Utimaco CryptoServer Key Storage Provider Device : 3001@127.0.0.1 Group : DE-ANH1 Mode : External Key Storage I: 1 keys deleted </pre>
---------------	--

5.3.5 ExportKey

This command exports a key into a file.

Syntax	cngtool [Provider=<provider_name>] [Machine] Name=<key_name> [Spec=<key_specifier>] [Type=<type>] [Password=<pass>] ExportKey[=<filename>]
---------------	--

Parameter	Description
<provider_name>	Default setting is Utimaco CryptoServer Key Storage Provider, optional. If the parameter <code>Provider=</code> is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed. The set of valid provider names is retrieved by executing <code>cngtool EnumProvider</code> .
Machine	This optional parameter defines a machine-specific keyset instead of a user-specific one, optional. It is ignored by Utimaco's CSP/CNG Provider.

Parameter	Description
Name=<key_name>	Specifies the name of the key to be exported
Spec=<key_specifier>	Optional. Default value is 0.
<type>	<p>Defines the type of key blob to be exported. Default setting is a file in PKCS8 format. Possible values:</p> <ul style="list-style-type: none"> ▪ PKCS8 This type of key blob requires a password entry, i.e., Password=<pass> must be defined. This type of key blob is not supported for Utimaco's CSP/CNG Provider. ▪ P8 Same as PKCS8 ▪ private The resulting key blob is of type BCRYPT_PRIVATE_KEY_BLOB and contains both the private and public part of a key in an unencrypted fashion. This type of key blob ignores a password entry and requires an Export property (set on key creation) to allow_plain. ▪ public Resulting key blob is of type BCRYPT_PUBLIC_KEY_BLOB and contains the public part of a key only. This type of key blob ignores a password entry.
Password=<pass>	Optional password to additionally protect the key blob. Only required for key blobs of type PKCS8 or P8.
<filename>	Optional path and filename for the key blob. If no path is given, the file is created in the current directory. If no filename is given, a default filename is created from the key name, key specifier and the type of the key blob.

Example	ccngtool Name=DEMOrsa Type=public ExportKey
----------------	---

Example	<p>On successful execution of the command, a success message with the export location is returned. Example output: On success, for example:</p> <pre> ----- Provider : Utimaco CryptoServer Key Storage Provider Device : 3001@127.0.0.1 Group : DE-ANH1 Mode : External Key Storage -----I: Successfully exported key to: DEMOrsa_0.BCRYPT_PUBLIC_KEY_BLOB </pre>
----------------	--

5.3.6 ImportKey

This command is supported only for Utimaco's CryptoServer Key Storage Provider and imports a key blob from a file into a CryptoServer.

Syntax	<code>cngtool [Provider=<provider_name>] [Machine] Name=<key_name> [Export=<export>] [Usage=<usage>] [Spec=<key_specifier>] [Type=<type>] [Password=<pass>] ImportKey=<filename></code>
---------------	---

Parameter	Description
<provider_name>	Default setting is Utimaco CryptoServer Key Storage Provider, optional. If the parameter <code>Provider=</code> is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed. The set of valid provider names is retrieved by executing <code>cngtool EnumProvider</code> .
Machine	This optional parameter defines a machine-specific keyset instead of a user-specific one, optional. It is ignored by Utimaco's CSP/CNG Provider.
Name=<key name>	Specifies the name of the key to be imported
Export=<export>	<p>This parameter defines whether the key can be exported or not. If the export policy in the CNG configuration forbids a certain export type, the next possible lower type is chosen. For example, if <code>allow_plain</code> is requested but the configuration file denies plaintext export and allows encrypted export, <code>allow</code> is selected for the new key. Possible values:</p> <ul style="list-style-type: none"> <code>deny</code> - denies the export of the key in both encrypted form and in plaintext format <code>allow</code> - the key can be exported in encrypted form <code>allow_plain</code> - the key can be exported in encrypted or plaintext format <p>The Export parameter is expected to be set.</p>
Usage=<usage>	<p>This parameter describes for which cryptographic functions the key can be used. Possible values:</p> <ul style="list-style-type: none"> <code>decrypt</code> - decryption/encryption <code>sign</code> - signature generation <code>agree</code> - secret agreement <code>all</code> - decryption/encryption, signature generation, secret agreement <p>Multiple values can be given as comma separated list. The Usage parameter is expected to be set.</p>
Spec=<key_specifier>	Optional. Default value is 0.

Parameter	Description
<type>	<p>Defines the type of key blob to be imported. Default setting is a file in PKCS8 format. Possible values:</p> <ul style="list-style-type: none"> ▪ PKCS8 Expects a key blob file in PKCS8 format. If the file name has the file extension <code>.p12</code> or <code>.pfx</code>, a file in PKCS12 format is expected. This type of key blob requires a password entry, i.e., Password=<pass> must be defined. ▪ P8 Same as PKCS8 ▪ private Expects a key blob file in <code>BCRYPT_PRIVATE_KEY_BLOB</code> format. This type of key blob ignores a password entry. ▪ public Expects a key blob file in <code>BCRYPT_PUBLIC_KEY_BLOB</code> format. This type of key blob ignores a password entry.
Password=<pass>	Password to additionally protect the key blob. Only required for key blobs of type PKCS8 or PKCS12.
<filename>	Path and filename for the key blob to be imported. If no path is given, the specified file is expected to be in the current directory.

Example	<code>cngtool Name=DEMOrsa Type=public Export=allow Usage=decrypt,sign ImportKey=DEMOrsa_0.BCRYPT_PUBLIC_KEY_BLOB</code>
----------------	--

Example	<p>On successful execution of the command, a success message with the import location is returned.</p> <p>Example output:</p> <pre> ----- - Provider : Utimaco CryptoServer Key Storage Provider Device : 3001@127.0.0.1 Group : DE-ANH1 Mode : External Key Storage ----- -I: Successfully imported key from: DEMOrsa_0.BCRYPT_PUBLIC_KEY_BLOB </pre>
----------------	---

5.4 Backup and Restore Commands

The backup and restore functions implemented in the CNG key management tool apply only for Utimaco's CSP/CNG Provider.

The cryptographic keys stored in the key database of the CryptoServer can be backed up in a secure manner in an external key backup file (`*.kbk`).



The key backup files are encrypted and signed with the CryptoServer's Master Backup Key.

If the MBK in MBK slot 3 is the autogenerated MBK named `AUTO-GEN`, no backup can be performed. Import a different MBK into MBK slot 3 using the `csadm MBKImportKey` command described in *CryptoServer - csadm Manual*.

5.4.1 BackupKey

This command creates a backup copy of a key stored in the internal or external key storage of the CryptoServer CNG Key Storage Provider.



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

It is important to note down which MBK has been used because for a successful restoring of this backup file at a later date it is necessary that the same MBK is in MBK slot 3 or after an MBK rollover in an MBK slot ≥ 3 . Otherwise, the backup file is inaccessible. This might be the result of the execution of a `csadm MBKImportKey` command. See section *Master Backup Key Rollover* in the *CryptoServer - Administration Manual* for details. It is not possible to retrieve the MBK by which a backup file has been generated from this backup file.

Syntax

```
cngtool Name=<key_name> [Spec=<key_specifier>]
BackupKey[=<filename>]
```


Parameter	Description
Name=<key_name>	Specifies the name of the key to be backed up
Spec=<key_specifier>	Optional. Default value is 0.
<filename>	If not specified, by default a file with the file name format <key_name>_<key_specifier>.kbk is created in the directory where the CNG tool is started from.

Example	cngtool Name=RSAKey BackupKey
----------------	-------------------------------

Example	<p>On successful execution of the command, a success message with the backup location is returned.</p> <pre>----- - Provider : Utimaco CryptoServer Key Storage Provider Device : 3001@127.0.0.1 Group : DE-ANH1 Mode : Internal Key Storage ----- - I: Successfully backed up key to: RSAKey_0.kbk</pre>
----------------	---

5.4.2 RestoreKey

This command imports a key from a backup copy in *.kbk format into the internal or external key storage of the CryptoServer CNG Key Storage Provider.

A key backup file can be restored to any CryptoServer which contains the appropriate Master Backup Key. Thus, multiple CryptoServer can be synchronized (e.g. in order to contain the same CA's signature key).



If a key with the same Name and Spec is already available, it is overwritten.



If keys are stored externally, they can be backed up or restored alternatively by either copying the complete key directory or single key blob files.

Syntax

```
cngtool RestoreKey=<filename>
```

Parameter**Description**

<filename>

Filename for the key backup blob

Example

```
cngtool RestoreKey=RSAKey_0.kbk
```

Example

On successful execution of the command, a success message with the name of the used backup file is returned.

--

Provider : Utimaco CryptoServer Key Storage

Provider

Device : 3001@127.0.0.1

Group : DE-ANH1

Mode : Internal Key Storage

--I:

Successfully restored key from:

RSAKey_0.kbk

5.4.3 RecryptExternalKeys

This command is used when an MBK rollover is performed. See section *Master Backup Key Rollover* in the *CryptoServer - Administration Manual* for details about an MBK rollover.

The `cngtool RecryptExternalKeys` command works only with the Utimaco key storage provider. This command recrypts the cryptographic keys in the external CNG keystore with the current MBK (typically an AES MBK in MBK slot 3). Only those keys are encrypted that the current user has Windows access rights for. The old MBK that was used to encrypt the external keystore before must be loaded into another MBK slot (≥ 4).



The `cngtool RecryptExternalKeys` command only recrypts external cryptographic keys if the old MBK and the new MBK are AES MBKs.



We highly recommend to create a backup of the external keystore files before you perform the `cngtool RecryptExternalKeys` command.

Syntax

```
cngtool RecryptExternalKeys
```

Example

```
cngtool RecryptExternalKeys
```

Example

On successful execution of the command, a success message is returned.

```
-----  
-  
Provider           : Utimaco CryptoServer Key Storage  
Provider  
Device             : 3001@127.0.0.1  
Group              : CNG  
Mode               : External Key Storage  
-----  
-  
I: Recrypted 1 key(s), 0 failed.
```

5.5 Migration Commands

The commands described in this section are used only for migration purposes, and are therefore not listed in the list of `cngtool` commands displayed by the `cngtool Help` command. They are only available for Utimaco's CryptoServer Key Storage Provider 2.x and higher, which has been first provided with the SecurityServer/CryptoServer SDK product CD version 4.10.

5.5.1 CreateConfigFile

Syntax

```
cngtool CreateConfigFile=<filename>
```

Parameter	Description
<filename>	Path and filename of the CNG Provider configuration file The path, the filename and the file extension can be individually chosen. IMPORTANT is that they are manually set in the <code>CS_CNG_CFG</code> system environment variable.

Example	<code>cngtool CreateConfigFile=C:\ProgramData\Utimaco\CNG\cs_cng.cf</code>
----------------	--

5.5.2 MigratePermissions

Security permissions define who and in what way is granted access to the keys stored on the CryptoServer Key Storage Provider (see [Windows Access Control for Keys \(p. 10\)](#)).

This command migrates the Microsoft Windows security permissions from the registry to key properties. If a key permission property is already set, it is not overwritten and remains unchanged.

Syntax	<code>cngtool MigratePermissions</code>
---------------	---

Example	<code>cngtool MigratePermissions</code>
----------------	---

5.6 Managing CSP Containers with CNG tool

5.6.1 General Syntax Mapping

CSP containers include keypairs of key exchange and signature keys.

To manage CSP Containers with the commands in [Key Management Commands \(p. 30\)](#) and [Backup and Restore Commands \(p. 40\)](#), the key specifier must be set explicitly depending on the type of the keypairs in the container:

CSP Key type	Key specifier in CNG tool
AT_EXCHANGE	spec=1
AT_SIGNATURE	spec=2

CSP container names will be handled in CNG tool as key names.

5.6.1.1 Creating a CSP Container for Key Exchange Purposes

To create a CSP container for key exchange purposes (key algorithm "AT_EXCHANGE"), use `cngtool createkey` with the key specifier (spec) "1" and set usage to "sign", "decrypt" and "agree".

Example	<pre>cngtool name=CSPContainer spec=1 usage=sign,decrypt,agree createkey=RSA,2048 Result: cngtool name=CSPContainer spec=1 keyinfo</pre>
----------------	---

Output	<pre>Example output: ----- Provider : Utimaco CryptoServer Key Storage Provider Device : 3001@127.0.0.1 Group : CNG Mode : External Key Storage ----- 1.: AlgId : RSA Name : CSPContainer Spec : 00000001 Flags : 00000000 Size : 2048 Uname : 33D4AC63FEDAEADA8F46F4FA177BBC77 AlgoGroup : RSA Export : 0x00000001 [+ ALLOW] Usage : 0x00000037 [+ DECRYPT + SIGN + AGREEMENT] Type : 0x00000000 [User] Block Length : 0x00000100 SecurityDescriptor: 0:S-1-5-21-2966636039-209776147-1487739335-2916G:DUD: (A;;0xd01f0003;;;S-1-5-21-2966636039-209776147-1487739335-2916)</pre>
---------------	--

5.6.1.2 Creating a CSP Container for Signature Purposes

To create a CSP container for signature purposes (key algorithm "AT_SIGNATURE"), use `cngtool createkey` with the key specifier (spec) "2" and set usage to "sign".

Example	<pre>Cngtool name=CSPContainer spec=2 usage=sign createkey=RSA,2048 Result: cngtool name=CSPContainer spec=2 keyinfo</pre>
----------------	---

Output	Example output:
	<pre>----- Provider : Utimaco CryptoServer Key Storage Provider Device : 3001@127.0.0.1 Group : CNG Mode : External Key Storage ----- 1.: Algid : RSA Name : CSPContainer Spec : 00000002 Flags : 00000000 Size : 2048 Uname : 3188850FB78185F0D0DE7FE899F9AE4E AlgoGroup : RSA Export : 0x00000001 [+ ALLOW] Usage : 0x00000022 [+ SIGN] Type : 0x00000000 [User] Block Length : 0x00000100 SecurityDescriptor: 0:S-1-5-21-2966636039-209776147-1487739335-2916G:DUD: (A;;0xd01f0003;;;S-1-5-21-2966636039-209776147-1487739335-2916)</pre>

5.6.2 Additional Notes

Generally, it's not possible to filter CSP Container with the command `cngtool listkeys`. CSP Containers can be recognized by the specifier "spec" value.

Example	<code>cngtool listkeys</code>
----------------	-------------------------------

Output	Example output:				

	Provider		: Utimaco CryptoServer Key Storage Provider		
	Device		: 3001@127.0.0.1		
	Group		: CNG		
	Mode		: External Key Storage		

	Index	AlgId	Size	Group	
	Name			Spec	

1	RSA	2048	CNG	CSPContainer	
2					
2	RSA	2048	CNG		
CSPContainer					
1					
3	RSA	2048	CNG		
NonCSPContainer					
0					

6 References

Reference	Title/Company	Doc. No
[CSADMIN]	CryptoServer – csadm Manual / Utimaco IS GmbH.	2009-0003
[CSMSADM]	CryptoServer – Administration Manual / Utimaco IS GmbH.	M010-0001-en
[CNG]	Cryptography API: Next Generation – Microsoft 2007. Available: https://msdn.microsoft.com/en-us/library/aa376210(VS.85).aspx	