CryptoServer

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

Manual for System Administrators

utimaco®

Imprint

Copyright 2017	Utimaco IS GmbH Germanusstr. 4 D-52080 Aachen Germany
Phone	+49 (0)241 / 1696-200
Fax	+49 (0)241 / 1696-199
Internet	http://hsm.utimaco.com
e-mail	hsm@utimaco.com
Document Version	2.0.2
Date	February 2017
Status	Final
Document No.	2008-0002
All Rights reserved	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. 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. All trademarks and registered trademarks are the property of their respective owners.

Table of Contents

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 Document

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 conventions in this manual:

Convention	*Use*	*Example*
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press the **OK** button.
`Monospaced`	File names, folder and directory names, commands, file outputs, programming code samples	You will find the file `example.conf` in the `/exmp/demo/` directory.
Italic	References and important terms	See Chapter 3, "Sample Chapter" in the *CryptoServer LAN/CryptoServer CryptoServer Command-line Administration Tool -csadm -Manual for System Administrators* or [CSADMIN].

Table 1: Document conventions

We have used 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.

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, `cxi.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, `cxi.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.

Both CryptoServer CSP and CryptoServer CNG Key Storage Provider are supported on Windows 7/8/8.1 and Windows Server 2008/2008 R2/2012/2012 R2.

Please refer to Microsoft’s MSDN web pages for a detailed specification of the CSP/CNG functionality [CNG].

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 that 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 5.000 RSA keys (1024-bit) and 14.000 ECDSA keys (NIST-P256) can be stored inside the CryptoServer.

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	✓	✓

Algorithm	*Key/Hash length/Curves (bits)*	*Interface*	
		CSP	*CNG*
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-224	224	✓	✓
SHA-256	256	✓	✓
SHA-384	384	✓	✓
SHA-512	512	✓	✓

Table 2: List of supported key algorithms

2.3 Key Access Restriction

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 chapter 4.2 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'

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

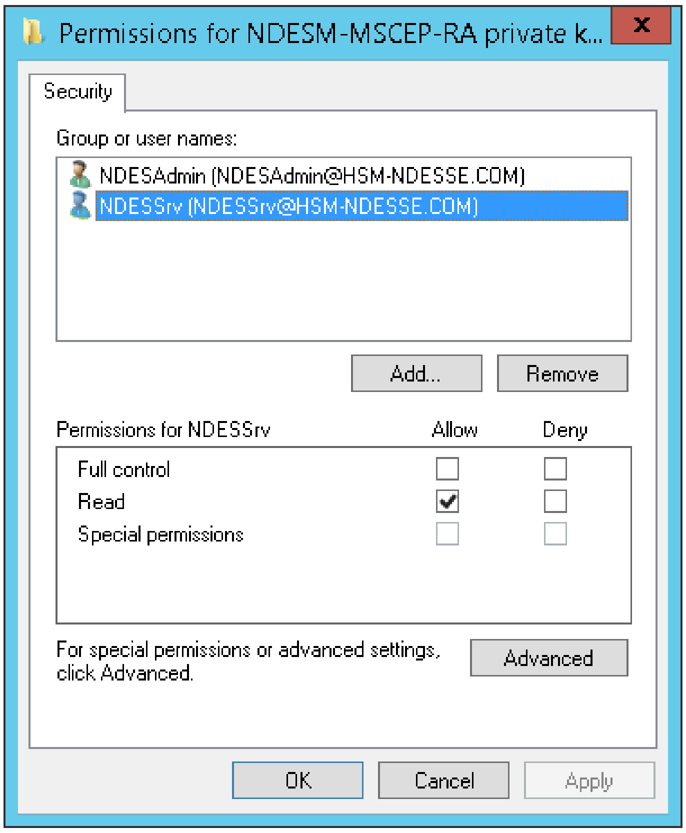
Table 3: Examples for defining key access restrictions

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, as also the corresponding cryptographic keys, in the internal key database of the CryptoServer, `CXI.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:


Permissions for NDESM-MSCEP-RA private k...

Security

Group or user names:

NDESAdmin (NDESAdmin@HSM-NDESSE.COM)
NDESSrv (NDESSrv@HSM-NDESSE.COM)

Permissions for NDESSrv	Allow	Deny
Full control	☐	☐
Read	✓	☐
Special permissions	☐	☐

For special permissions or advanced settings, click Advanced.

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.

Existing access rights for a container/key can be migrated as described in chapter 4.3.

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.

CSP Function	*Required Access Right*
`CryptAcquireContext`	■ Full control if `CRYPT_DELETEKEYSET` flag is set ■ Read unless the default container is to be opened or the `CRYPT_DELETEKEYSET` or `CRYPT_NEWKEYSET` flag is set
`CryptDecrypt`	Read if it is a persistent key (see `CryptGenKey`)
`CryptEncrypt`	Read if it is a persistent key (see `CryptGenKey`)
`CryptExportKey`	Read if it is a persistent key (see `CryptGenKey`). 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 `AT_KEYEXCHANGE` or `AT_SIGNATURE` key is to be created and `CRYPT_VOLATILE` flag is not set, i.e. for persistent keys only
`CryptGetKeyParam`	Read if it is a persistent key (see `CryptGenKey`)
`CryptGetProvParam`	Read if `PP_KEYSET_SEC_DESCR` is queried
`CryptGetUserKey`	Read
`CryptHashSessionKey`	Read if it is a persistent key (see `CryptGenKey`)
`CryptImportKey`	Full control if a Utimaco Backup BLOB is to be imported Full control for import container if a `AT_KEYEXCHANGE` or `AT_SIGNATURE` key is to be imported and the `CRYPT_VOLATILE` 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 `CryptGenKey`)
`CryptSetProvParam`	Full control if `PP_KEYSET_SEC_DESCR` is to be set
`CryptSignHash`	Read
`CryptVerifySignature`	Read if it is a persistent key (see `CryptGenKey`)

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.

CNG Function	*Required Access Right*
`NCryptCreatePersistedKey`	Full control if the key exists and the `NCRYPT_OVERWRITE_KEY_FLAG` is set. Check is done on `NCryptFinalizeKey`.
`NCryptDecrypt`	Read
`NCryptDeleteKey`	Full control
`NCryptEncrypt`	Read
`NCryptExportKey`	Read
`NCryptFinalizeKey`	See `NCryptCreatePersistedKey` and `NCryptImportKey`
`NCryptGetProperty`	Read if used with a key handle
`NCryptImportKey`	Full control for a key to be imported if the key exists and the `NCRYPT_OVERWRITE_KEY_FLAG` is set; read for import key. Check is done on `NCryptFinalizeKey` if `NCRYPT_DO_NOT_FINALIZE_FLAG` is set.
`NCryptOpenKey`	Read

CNG Function	*Required Access Right*
`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

2.5 Key Management

Utimaco also provides appropriate tools for managing cryptographic keys. These can be found on the SecurityServer and the CryptoServer SDK product CDs in the folder `\Software\Windows\<x86-xx>\Crypto_APIs\CSP-CNG\bin\`.

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

Both tools are installed by the CryptoServer host-software installer when selecting the installation of the CSP/CNG – Utimaco CryptoServer Key Storage Provider. By default, the installation folder, which is automatically added to the PATH environment variable is:

- For SecurityServer: `C:\Program Files\Utimaco\CryptoServer\Administration\` or
- For CryptoServer SDK - `C:\Utimaco\CryptoServer\Administration\`.

Before you start using the CSP Tool, you shall configure the CSP/CNG Provider as explained in chapter 4. To start the CSP Tool, select Windows Start Menu > All Programs > Utimaco > CryptoServer > CSP Tool.

To start using the CNG tool just open a command prompt and type `cngtool <command>`. For a list of available commands see chapter 5 in this manual.

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 PCI/PCIe plug-in card integrated 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 (`CS-Series`, `CSe-Series, Se-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 [CSMSADM] or [CSADMIN] for help.
- The CryptoServer host-software is installed.

  The CryptoServer installer software, `CryptoServerSetup-X.XX.X.X.exe`, 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 4.2.

4 Configuration

This chapter shows how to configure the CSP/CNG Provider 1.x with the CSP/CNG Control Applet, which was provided with the SecurityServer/CryptoServer SDK product CD 4.01 and earlier.

As from SecurityServer/CryptoServer SDK 4.10 the CSP/CNG Control Applet is replaced by the CNG configuration file which is described in chapter 4.2 and is used for configuring the CSP/CNG Provider 2.x.

Here you also find instructions on how to migrate your software from CSP/CNG Provider 1.x to CSP/CNG Provider 2.x.

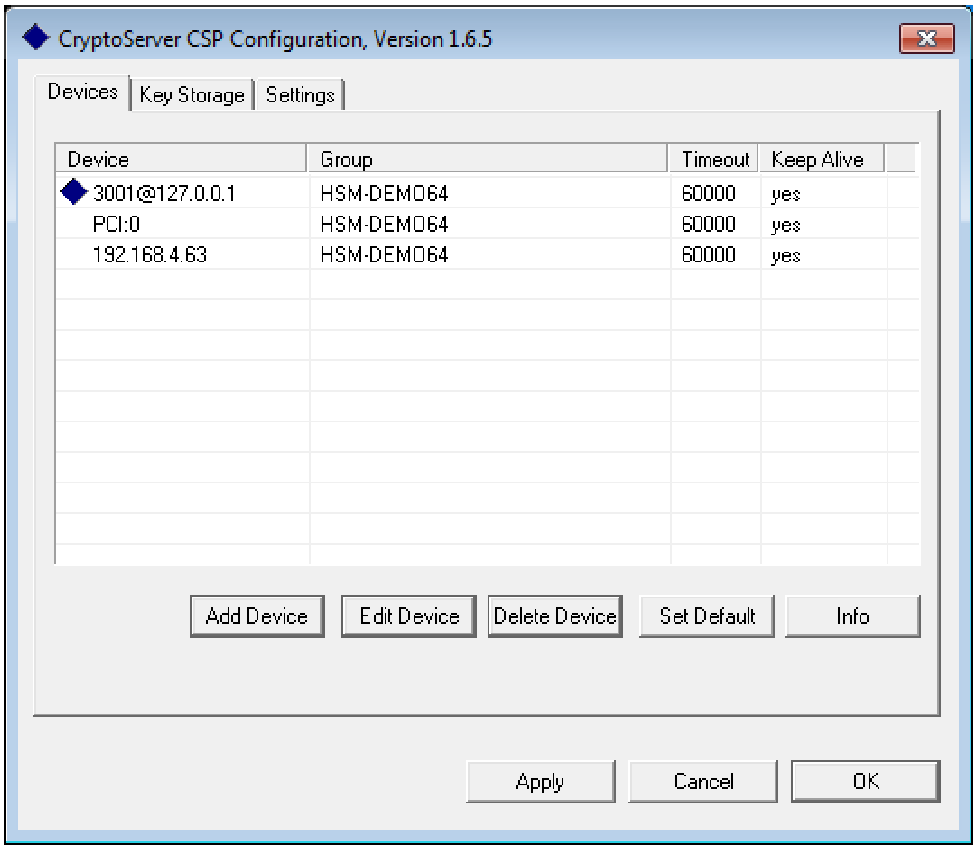
4.1 Using the CSP/CNG Control Panel Applet

The control panel applet is only part of SecurityServer before release 4.10 and is not supported anymore starting with SecurityServer 4.10.

Utimaco's CSP/CNG Provider 1.x can be configured by using the CSP/CNG Control Panel Applet.

*If you are using a Windows 7 operating system or higher you have to start the application with administrator rights. To do that, click with the right-hand mouse button on **CSP Configuration**. Select the option **Run as Administrator** and accept the security warning with **Yes**.*

1. Click **Start** > **All Programs** > **Utimaco** > **CryptoServer** > **CSP Configuration** to start the control panel applet.

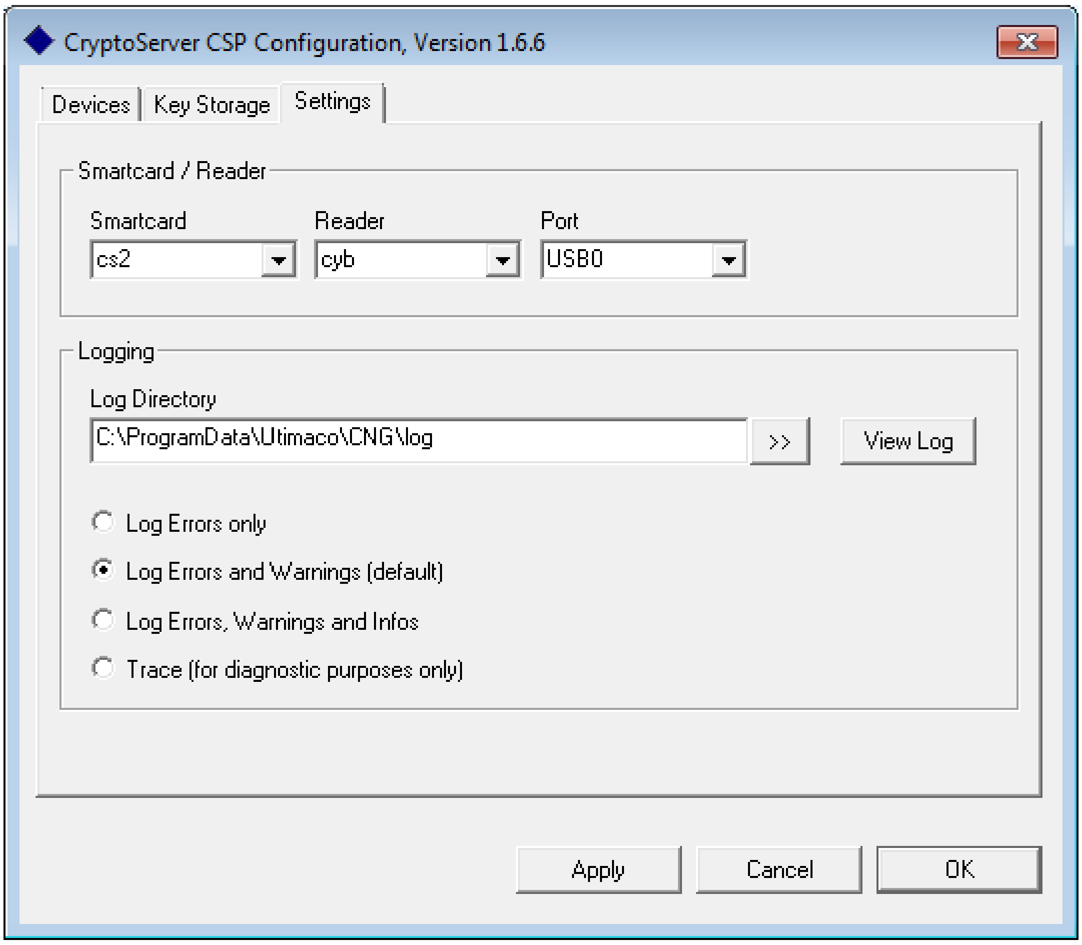


Device	Group	Timeout	Keep Alive
◆ 3001@127.0.0.1	HSM-DEMO64	60000	yes
PCI:0	HSM-DEMO64	60000	yes
192.168.4.63	HSM-DEMO64	60000	yes

The **Devices** tab shows the list of configured CryptoServer devices. This can be either a local CryptoServer PCI/PCIe plug-in card or a CryptoServer LAN appliance (via TCP) or the CryptoServer Simulator. The blue rhombus marks the default device, which will be used exclusively in case of internal key storage or preferred in case of external key storage.

A click on the **Apply** button writes all configuration data to the registry without closing the dialog, the **OK** button does the same but closes the dialog box, the **Cancel** button closes the dialog box without saving modifications.

2. Click the **Settings** tab.



Smartcard / Reader

Smartcard: cs2

Reader: cyb

Port: USB0

Logging

Log Directory: C:\ProgramData\Utimaco\CNG\log

Log Errors only

[✓ Log Errors and Warnings (default)]

Log Errors, Warnings and Infos

Trace (for diagnostic purposes only)

3. In the configuration area **Smartcard / Reader**, select the source for the PIN pad if the user authenticating the device configuration uses a smartcard as storage medium for his authentication key. For the USB PIN pad REINERSCT cyberJack delivered by Utimaco the correct settings are shown in the figure above.

   The following smartcard readers are supported:

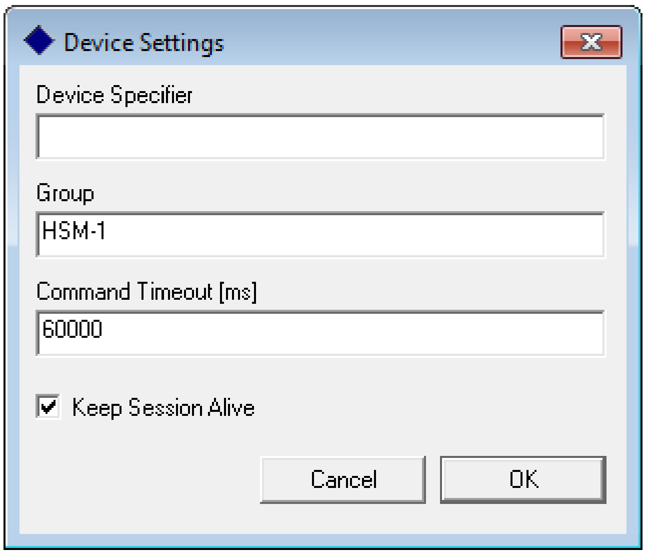
Specifier	*Model / Vendor*	*Type*
cyb	cyberJack / REINERSCT	serial / USB
cp8	XiMax / Xiring	serial
acr	ACR80 / Advanced Card Systems	serial
pcsc	any reader supporting PCSC	serial / USB

   Depending on the PIN pad connection type a dedicated port (e.g. COM3) or the position within an enumeration (e.g. USB:0) has to be chosen.

Here, you can additionally set the location where Utimaco's *CryptoServer CNG Key Storage Provider* writes the log file `cs2cng.log` (default is `%AllUserProfile%\Utimaco\CNG\log`). The file automatically rotates, if the maximum file size of 8 MByte has been reached: In this case the current log file is renamed to `cs2cng.log.bak` and a new, empty log file is created. Depending on the chosen log level, errors, warnings and informational messages are written to the log file.

We recommend you to set the log level to `errors and warnings`, and increase it only in case of errors.

4. Click the **Add Device** button.



Device Specifier

Group: HSM-1

Command Timeout [ms]: 60000

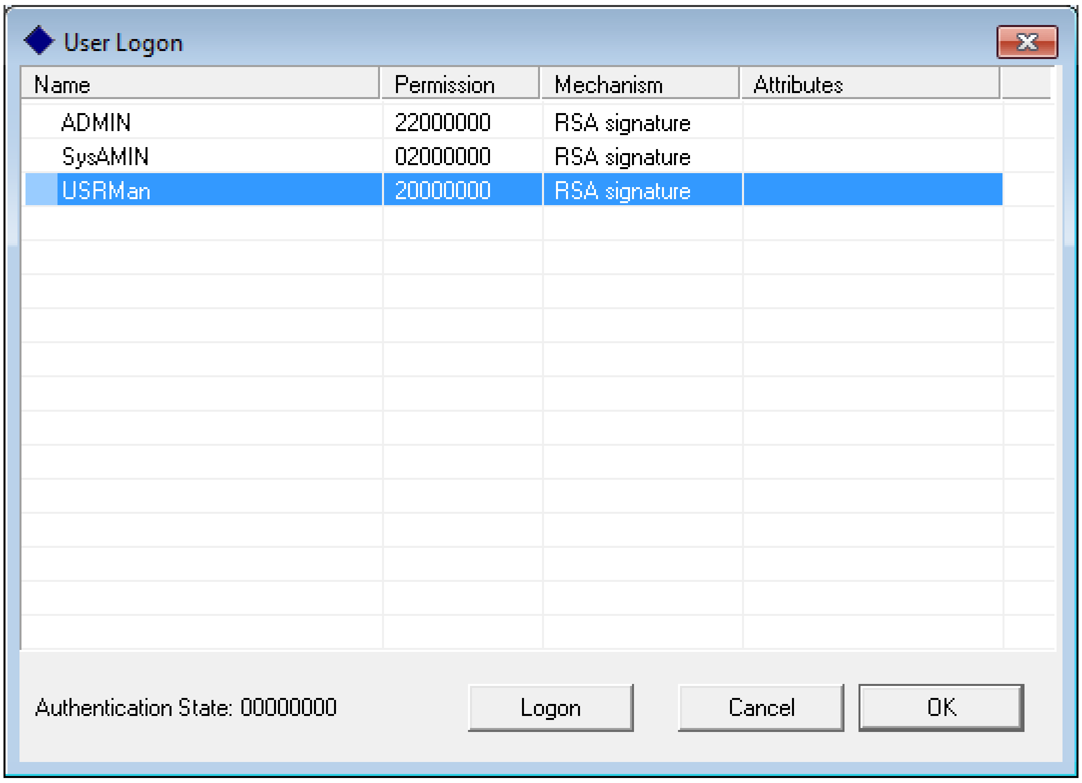
[✓] Keep Session Alive

5. Add a **Device Specifier** for your CryptoServer device.
   - If this is a CryptoServer PCI or PCIe plug-in card, this is `PCI:0`.
   - If this is a CryptoServer LAN, enter an IPv4 or IPv6 address (for example `192.168.5.17`).
   - If you want to use the CryptoServer Simulator, enter `3001@127.0.0.1` as the device address.

   Under **Group** you see the name of a user on the host computer where the CryptoServer CSP configuration application is installed. This name is used as the name of the user (permissions: 00000002) who is automatically created on the CryptoServer once you have successfully finished the device configuration. This new user is used to authenticate the management and use of cryptographic keys generated by the CryptoServer.

The **Group** name defines the group to which all generated cryptographic keys belong to. In order to strictly restrict the access to the cryptographic keys only users who are members of the same **Group** as the key are permitted to manage and use it. The automatically generated user mentioned above, belongs by default to this **Group**.

6. Keep the suggested **Group** name or enter a different one.
7. Click the **OK** button.
8. Select in the **User Logon** dialog box the ADMIN user or a user with sufficient permissions in the user group 7 (min. 20000000).

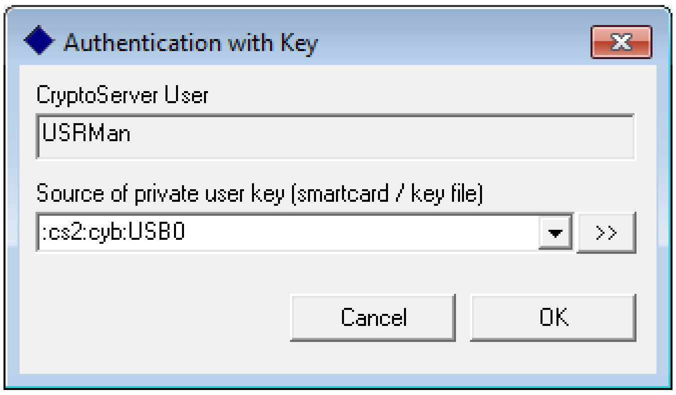


User Logon

Name	Permission	Mechanism	Attributes
ADMIN	22000000	RSA signature	
SysAMIN	02000000	RSA signature	
USRMan	20000000	RSA signature	

Authentication State: 00000000

9. Click the **Logon** button.



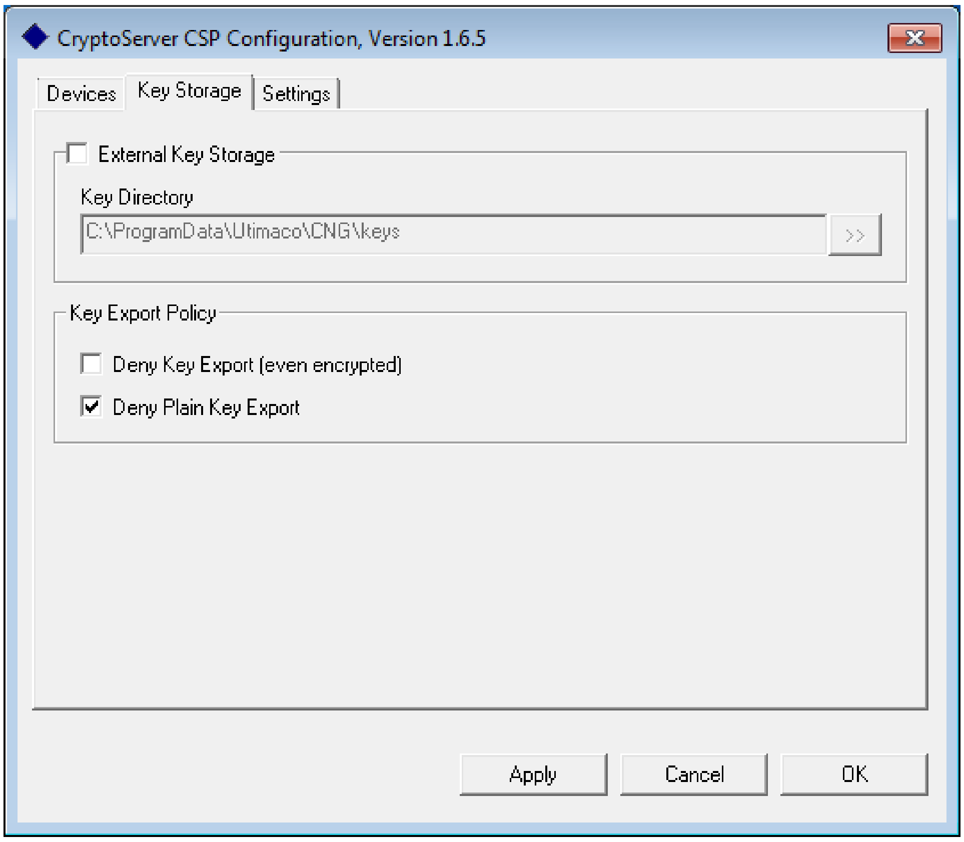
Authentication with Key

CryptoServer User: USRMan

Source of private user key (smartcard / key file): :cs2:cyb:USB0

- If the user is using an RSA Signature stored on a smartcard for authentication, click **OK** and follow the instructions on the display of the PIN pad.
- If the user is using an RSA Signature stored in a keyfile for authentication, click the **>>** button, select the keyfile storage location and double-click the corresponding keyfile. Then click **OK**.

10. Click **OK** in the dialog box **User Logon**. The dialog box closes.
11. Click the **Apply** button.
12. Click the **Key Storage** tab.



- [ ] External Key Storage

Key Directory: C:\ProgramData\Utimaco\CNG\keys

Key Export Policy
- [ ] Deny Key Export (even encrypted)
- [x] Deny Plain Key Export

On the **Key Storage** tab the key storage location and the key policy can be defined.

On selection of external key storage the keys are stored locally as key files (encrypted with the CryptoServer's Master Backup Key). A specific key directory may be chosen (default is `%AllUserProfile%\Utimaco\CNG\keys`). If external key storage is deselected, keys are stored internally on the CryptoServer.

The key export property of keys being created or imported can be restricted by a global policy. Basically, the key export property is set on generation or import of keys according to the parameters given by the application (e.g. Microsoft CA). If key export is restricted by the given policy, the application parameters (`allow key export`, `allow plain key export`) are overwritten respectively.

4.2 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.

4.2.1 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\Utimaco\CNG\cs_cng.cfg`

The `ProgramData` folder 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.

4.2.2 Configuration File Options

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

Parameter	*Description*
`Logging`	Specifies the log level. The following values are valid: ■ 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 IMPORTANT: Do not use log level higher than 2 in production environments.

Parameter	*Description*
`Logpath`	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 `cs2cng.log` is appended automatically by the CSP/CNG Provider.
`Logsize`	Defines the maximum size of the log file. If the maximum is reached, the old log file will be renamed to `cs2cng.log.01` and a new log file `cs2cng.log` is created. If a file `cs2cng.log.01` already exists, this file will be renamed to `cs2cng.log.02` and so on. Nine backup log files are kept at maximum. The file size can be defined as a value in bytes or as a formatted text. Valid text formats are: `kb`, `mb`, `gb` Examples: `Logsize = 1000` means that the log size is 1000 bytes. `Logsize = 1000kb` means that the log size is 1000 kilobytes.
`KeysExternal`	Specifies the type of the key storage: internal or external. This setting is of type Boolean. If `KeysExternal = true`, 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. If `KeysExternal = false`, the keys are only read/written from/to the internal key database of the CryptoServer.
`KeyStore`	Specifies the path to the external key storage (default `C:\ProgramData\Utimaco\CNG\keys`). This parameter shall be set if `KeysExternal = true`. 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.

Parameter	*Description*
`ExportPolicy`	Defines which export properties cannot be set by a CNG user. ■ 0: ▣ For CNG, the key export policy is defined by the `NCRYPT_ALLOW_EXPORT_FLAG` and the `NCRYPT_ALLOW_PLAINTEXT_EXPORT_FLAG` in a `NCryptSetProperty()` function call ▣ CSP keys are exportable as plaintext if created with the `CRYPT_EXPORTABLE` flag. ■ 1: ▣ For CNG, this parameter denies the plaintext export of newly generated keys by overruling the `NCRYPT_ALLOW_PLAINTEXT_EXPORT_FLAG` in a `NCryptSetProperty()` function call. ▣ CSP keys created with the `CRYPT_EXPORTABLE` flag can be exported when wrapped with another key. ■ 2: ▣ For CNG, this parameter denies both encrypted and plaintext export of newly generated keys by overruling both the `NCRYPT_ALLOW_EXPORT_FLAG` and the `NCRYPT_ALLOW_PLAINTEXT_EXPORT_FLAG` in a `NCryptSetProperty()` function call. ▣ For CSP keys the `CRYPT_EXPORTABLE` flag will be overruled and key export is denied completely.
`Group`	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.
`ConnectionTimeout`	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.
`CommandTimeout`	Specifies the maximum time in milliseconds to wait for the answer from CryptoServer after sending a command. Default value is 60000 ms.
`KeepAlive`	Keep sessions alive and prevent them from expiring after 15 minutes idle time. This setting is of type Boolean. Default value is `true`.

Parameter	*Description*
`Device`	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: ■ `Device=PCI:0` for a CryptoServer PCIe plug-in card ■ `Device=3001@127.0.0.1` for the CryptoServer Simulator ■ `Device=<IPv4 or IPv6 address> for a CryptoServer LAN appliance`
`Login`	Specifies the authentication credentials for the CryptoServer CNG-user, who is the only user permitted to generate/access cryptographic keys. At 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 permissions 00000002, attribute `CXI_GROUP=<Group>` and one of the authentication mechanisms listed in chapter 4.2.3. An example with csadm: `csadm Dev=PCI:0 LogonSign=UserAdmin,:cs2:cyb:USB0 AddUser=CNGUser,002{CXI_GROUP=HSM-1},hmacpwd,sma,ask` If the `Login` entry is not present in the configuration file, the CSP/CNG provider tries to display an interactive login dialogue. For Windows services (e.g., AD CS), this is only possible if the registry key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Windows\NoInteractiveServices` is set to value `0`.

4.2.3 Authentication Mechanisms

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.

- HMAC password:

```
Login = "username,password"
```

- RSA Signature stored in a keyfile:

```
Login = "username,RSASign=filename#password"
```

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

- RSA Signature with a keyfile stored on a smartcard; the PIN pad for reading the content of the smartcard is connected to the host computer:

```
Login = "username,RSASign=key-specifier#PIN"
```

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.

- RSA Signature with a keyfile stored on a smartcard; the PIN pad for reading the content of the smartcard is directly connected to the CryptoServer:

```
Login = "username,RSASC=key-specifier#PIN"
```

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.

- ECDSA Signature stored in a keyfile:

```
Login = "username,ECDSA=filename#password"
```

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

- ECDSA Signature with a keyfile stored on a smartcard; the PIN pad for reading the content of the smartcard is connected to the host computer:

```
Login = "username,ECDSA=key-specifier#PIN"
```

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.

4.3 Migration from CSP/CNG 1.x to CSP/CNG 2.x

When using the installer provided on the SecurityServer or CryptoServer SDK product CD 4.10 or later, the first part of the migration is automatically performed. In case of an error, the migration can be manually performed with the CNG tool as follows:

1. Check that the `CS_CNG_CFG` environment variable points to the intended configuration file, for example `C:\ProgramData\Utimaco\CNG\cs_cng.cfg`.
2. Export the CSP/CNG configuration data available in the registry into the configuration file defined in the environment variable `CS_CNG_CFG`.

```
cngtool CreateConfigFile=%CS_CNG_CFG%
```

If the CSP/CNG 1.x provider has been used in a failover scenario, that means, multiple devices were configured, the configuration has to be changed manually. The CSP/CNG 2.x provider requires one and the same login credentials for all devices and only those of the primary device are migrated automatically. Therefore, one the same user, i.e., with exactly the same login credentials, has to be created on all devices and provided in the CSP/CNG configuration file.

If Windows key access permissions have been set previously, they have to be migrated to internal key permissions. In order to do so, the Utimaco CSP/CNG provider 2.x needs to be installed and configured correctly on the host system.

Utimaco's CSP/CNG Provider 2.0 provided on the SecurityServer/CryptoServer SDK product CD 4.10 requires the CXI firmware module version 2.2.0.0 or later to be loaded on the CryptoServer.

Execute the following command to migrate the permissions:

```
cngtool MigratePermissions
```

Note that permissions for a key are set only once: If permissions for a key have been set on multiple hosts, only the permissions on the host, where the CNG tool command `MigratePermissions` has been executed first, are migrated. If existing permissions are found on further command executions, a message is printed and the user has to check and adjust the key permissions manually.

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,256"

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 is given as a parameter
`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 provider and the maximum length, in characters, of the name of a persistent key
`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 new key of type RSA or ECDSA in the internal or external key database of the specific Key Storage Provider.
`DeleteKey`	Delete a specified key of type RSA or ECDSA form the internal or external key database of the specific Key Storage Provider.
`ExportKey`	Export a Microsoft key blob into a file

Command	*Description*
`ImportKey`	Import a Microsoft key blob from a file
Backup/Restore Commands	
`BackupKey`	Create a backup copy of a cryptographic key internally stored in the database of the CryptoServer CNG Key Storage Provider protected with the Master Backup Key of the CryptoServer.
`RestoreKey`	Restore a cryptographic key from a backup copy into the internal key database of the CryptoServer CNG Key Storage Provider.

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	`cngtool Help`
	`cngtool Help=<command>`

Parameter	`<command>` specific cngtool command
Example	`cngtool Help=ListKeys`

Output	`List keys stored on Key Storage Provider`
	`syntax:`
	`   cngtool [Provider=<provider_name>] [Machine] [Name=<key_name>] ListKeys`

5.2.2 EnumProvider

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

Syntax	`cngtool EnumProvider`
Example	`cngtool EnumProvider`
Output	`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	`cngtool [Provider=<provider_name>] ProviderInfo`
Parameter	`<provider_name>` Default setting is Utimaco CryptoServer Key Storage Provider. If the parameter `Provider=` is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed. The set of valid provider names is retrieved by executing `cngtool EnumProvider`.
Example	`cngtool ProviderInfo`
Output	see below

```
------------------------------------------------------------

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
```

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

```
cngtool [Provider=<provider_name>] ListAlgos
```

Parameter

`<provider_name>`
Default setting is Utimaco CryptoServer Key Storage Provider.

If the parameter `Provider=` is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed.

The set of valid provider names is retrieved by executing
`cngtool EnumProvider`.

Example

```
cngtool Provider="Utimaco CryptoServer Key Storage Provider"
ListAlgos
```

Output

Example output:

```
-------------------------------------------------------------


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
...
```

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

```
cngtool [Provider=<provider_name>] [Machine] [Name=<pattern>]
ListKeys
```

Parameter

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

```
cngtool Name=EXMP* ListKeys
```

Output

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
```

5.3.2 KeyInfo

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

Syntax

```
cngtool [Provider=<provider_name>] [Machine] Name=<key_name>
[Spec=<key_specifier>] KeyInfo
```

Parameter

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

Example

```
cngtool Name=EXMPrsa KeyInfo
```

Output

Example output:

```
------------------------------------------------------------


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
```

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

- `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
- `Mode` – type of the used key storage: external or internal (see also chapter 2.1).
- `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 chapter 4.2.2.
- `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

5.3.3 CreateKey

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

Syntax	`cngtool [Provider=<provider_name>] [Machine] [Export=<export>] [Usage=<usage>] Name=<key_name> [Spec=<key_specifier>] CreateKey=<key_type>,<bit_size>`
Parameter	■ `<provider_name>` (optional) Default setting is Utimaco CryptoServer Key Storage Provider. If the parameter `Provider=` is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed. The set of valid provider names is retrieved by executing `cngtool EnumProvider`. ■ `Machine` (optional)

This parameter defines a machine specific keyset instead of a user specific one. It is ignored by the Utimaco CSP/CNG Provider.

- `Export=<export>` (optional)
  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 `allow_plain` is requested but the configuration file denies plaintext export and allows encrypted export, `allow` is selected for the new key. Possible values:
  - `deny -` denies the export of the key in both encrypted form and in plaintext format
  - `allow` – the key can be exported in encrypted form
  - `allow_plain -` the key can be exported in encrypted or plaintext format

If the `Export` property is omitted, `allow` is set by default.

- `Usage=<usage>` (optional)
  This parameter describes for which cryptographic functions the key can be used. Possible values:
  - `decrypt -` decryption/encryption
  - `sign -` signature generation
  - `agree -` secret agreement
  - `all –` decryption/encryption, signature generation, secret agreement

Multiple values can be given as comma separated list.

Default setting for an RSA key (if Usage=<usage> is omitted) is `decrypt`, `sign`.

Default setting for an ECDSA key is `all`.

- `Name=<key_name>`
  Specifies the name of the key
- `Spec=<key_specifier>` (optional)
  Default value is 0.
- `<key type>`
  - `RSA`
  - `ECDSA`
- `<bit_size>`
  key size in bits
  - For RSA – 512 to 16.384 (delta = 8 bit)
  - For ECDSA – 256, 384, 521

Example

```
cngtool Export=allow Name=DEMOecdsa CreateKey=ECDSA,521
```

Output	On success, for example:

```
------------------------------------------------------------


Provider           : Utimaco CryptoServer Key Storage Provider
Device             : 3001@127.0.0.1
Group              : DE-ANH1
Mode               : External Key Storage


------------------------------------------------------------
```

Otherwise, an appropriate error message.

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,256"

5.3.4 DeleteKey

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

Syntax	`cngtool [Provider=<provider_name>] [Machine] Name=<pattern> [Spec=<key_specifier>] DeleteKey`
Parameter	■ `<provider_name>` (optional) Default setting is Utimaco CryptoServer Key Storage Provider. If the parameter `Provider=` is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed. The set of valid provider names is retrieved by executing `cngtool EnumProvider`. ■ `Machine` (optional) This parameter defines a machine specific keyset instead of a user specific one. 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.

Example

```
cngtool Name=DEMOecdsa DeleteKey
```

Output

On success, for example:

```
------------------------------------------------------------

Provider          : Utimaco CryptoServer Key Storage Provider
Device            : 3001@127.0.0.1
Group             : DE-ANH1
Mode              : External Key Storage
______________________________________________________________
I: 1 keys deleted
```

Otherwise, an appropriate error message.

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

- `<provider_name>` (optional)
  Default setting is Utimaco CryptoServer Key Storage Provider.
  If the parameter `Provider=` is omitted, information about Utimaco's CryptoServer Key Storage Provider is displayed.
  The set of valid provider names is retrieved by executing `cngtool EnumProvider`.
- `Machine` (optional)
  This parameter defines a machine specific keyset instead of a user specific one.
  It is ignored by Utimaco's CSP/CNG Provider.
- `Name=<key_name>`
  Specifies the name of the key to be exported
- `Spec=<key_specifier>` (optional)
  Default value is 0.
- `Type=<type>`
  Defines the type of key blob to be exported. Default setting is a file in PKCS8 format.
  Possible values:
  - 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)
  Filename for the key blob. If no filename is given, a default filename is created from the key name, key specifier and the type of the key blob.

Example

```
cngtool Name=DEMOrsa Type=public ExportKey
```

Output

On success, for example:

```
------------------------------------------------------------


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
```

Otherwise, an appropriate error message.

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

```
cngtool Name=<key_name> [Spec=<key_specifier>] [Type=<type>]
[Password=<pass>] ImportKey=<filename>
```

Parameter

- `Name=<key_name>`
  Specifies the name of the key to be imported
- `Spec=<key_specifier>` (optional)
  Default value is 0.
- `Type=<type>`
  Defines the type of key blob to be imported. If this setting is omitted, by default a file in PKCS format is exported.
  Possible values:
  - PKCS8
    Expects a key blob file in PKCS8 format. If the file name has the file extension `.p12` or `.pfx,` 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 BCRYPT_PRIVATE_KEY_BLOB format.
    This type of key blob ignores a password entry.
  - public
    Expects a key blob file in BCRYPT_PUBLIC_KEY_BLOB format.
    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 `PKCS12`.
- `<filename>`
  Filename for the key blob to be imported.

Example

```
cngtool Name=DEMOrsa Type=public ImportKey
```

Output

On success, for example:

```
------------------------------------------------------------

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
```

	Otherwise, an appropriate error message.

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.

5.4.1 BackupKey

This command creates a backup copy of a key stored in the internal key database of the CryptoServer.

Syntax	`cngtool Name=<key_name> [Spec=<key_specifier>] BackupKey[=<filename>]`
Parameter	■ `Name=<key_name>` Specifies the name of the key to be backed up ■ `Spec=<key_specifier>` (optional) Default value is 0. ■ `<filename>` (optional) If not specified, by default a file with the file name format `<key_name>_<key_specifier>.kbk` is created in the folder where the CNG tool is started from.
Example	`cngtool Name=RSAKey BackupKey`
Output	On success, for example:

```
------------------------------------------------------------


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
```

Otherwise, an appropriate error message.

5.4.2 RestoreKey

This command imports a key from a backup copy in `*.kbk` format into the CryptoServer.

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).

Syntax

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

Parameter

`<filename>`
Filename for the key backup blob

Example

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

Output

On success, for example:

```
------------------------------------------------------------


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
```

Otherwise, an appropriate error message.

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.

5.5 Migration Commands

The commands described in this chapter 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 is first provided with the SecurityServer/CryptoServer SDK product CD version 4.10.

5.5.1 CreateConfigFile

This command is used to automatically convert the configuration previously set via the CSP Control Panel Applet and stored in registry keys into a configuration file for Utimaco's CSP/CNG Provider. For the configuration file to become effective, the path and its file name shall be set as the system environment variable `CS_CNG_CFG`. See chapter 4.2 for further details.

Syntax	`cngtool CreateConfigFile=<filename>`
Parameter	`<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 `CS_CNG_CFG` system environment variable.
Example	`cngtool CreateConfigFile=C:\ProgramData\Utimaco\CNG\cs_cng.cfg`

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 chapter 2.4).

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	`cngtool MigratePermissions`
Example	`cngtool MigratePermissions`

References

Reference	*Title/Company*	*Document No.*
[CSADMIN]	CryptoServer LAN/CryptoServer - CryptoServer Command-line Administration Tool – csadm – Manual for System Administrators/Utimaco IS GmbH.	2009-0003
[CSMSADM]	CryptoServer Manual for System Administrators/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	