CryptoServer

PKCS#11 p11tool2

Reference Manual

utimaco®

Imprint

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 https://support.hsm.utimaco.com/

e-mail support@utimaco.com

Document Version 1.6.7

Date 2023-04-20

Status **PUBLISHED**

Document No. 2012-0004

Table of Contents

1 Introduction

Thank you for purchasing our CryptoServer security system. We hope you are satisfied with our product. Please do not hesitate to contact us if you have any complaints or other comments.

1.1 About This Document

This document provides detailed description of the CryptoServer PKCS#11 Administration Tool Release 2 (p11tool2) commands.

When implementing your own PKCS#11 applications, please read also [CS_PKCS11DEV (p. 99)] and [CS_PKCS11HON (p. 99)] provided on the SecurityServer product CD and on the CryptoServer SDK CD under `\Documentation\Crypto_APIs\PKCS11_R3`.

1.1.1 Document Conventions

We use the following document conventions:

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

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 The CryptoServer PKCS#11 Administration Tool Release 2

p11tool2 is the CryptoServer PKCS#11 Administration Tool Release 2 based on the CryptoServer PKCS#11 Library R3. It is a command line utility designed for being called from the command line or in a batch file. The p11tool2 offers functions to execute PKCS#11 commands on the CryptoServer and additional commands for backup, restoration and configuration settings.

This document gives detailed command descriptions of the p11tool2. For further information about requirements and configuration, see [CS_PKCS11DEV (p. 99)].

p11tool2 is intended to be used for PKCS#11 slot management and key management on a single CryptoServer device. It cannot be used for the PKCS#11 management of a CryptoServer failover or load balancing cluster containing several CryptoServer devices.

2.1 Overview

The following table gives a short overview of the p11tool2 commands.

Certain types of shell processes treat certain characters (for example, commas, colons, semi-colons) differently. If the execution of a p11tool2 command fails with an error message from the shell about a missing parameter or an illegal parameter format, quoting parameter values may be necessary.

The following is an example for a correct p11tool2 command syntax in a Microsoft PowerShell:

```
p11tool2 [Slot=<slot_id>] Login="<user_name>,<auth_token>" <command>
```

Command	***Description***
Basic Commands:	
PKCS#11 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
`PrintError=`	Display the corresponding error message text to an error code.
`Version`	Show the version number of the p11tool2
`ListSlots[=]`	List all slots
`GetInfo`	Get general information about Cryptoki
`GetSlotInfo`	Get information about a specific slot
`GetTokenInfo`	Get information about a specific token
`LoginSO=`	Login as security officer (SO)
`LoginUser=`	Login as normal user
`Login=`	Login as generic user
`InitToken=`	Initialize a token
`InitPIN=`	Initialize the normal user PIN
`SetPIN=`	Change the PIN of the SO or the normal user

Command	***Description***
Basic Commands:	
PKCS#11 Commands:	
`ListObjects`	List available objects
`DeleteObject`	Delete a specific object
`ImportP12=`	Import certificate, public and private key form PKCS#12 file
`ImportCert=`	Import certificate and public key from certificate file
`ExportCert[=]`	Write the BER-encoding of a specific certificate to a file or to the standard output if no file name is set
`ExportP10=`	Export a certificate signing request
`GenerateKey=`	Generate a secret key or set of domain parameters
`GenerateKeyPair=`	Generate a public/private key pair
Configuration Commands:	
`ListConfig=`	List all configuration attributes
`GetLocalConfig=`	Get local configuration value
`GetGlobalConfig=`	Get global configuration value
`GetSlotConfig=`	Get slot configuration value
`SetGlobalConfig=`	Set global configuration value
`SetSlotConfig=`	Set slot configuration value

Command	***Description***
Basic Commands:	
PKCS#11 Commands:	
Backup/Restore Commands:	
`GetBackupInfo`	Get information about the given backup key
`BackupInternalKeys=`	Backup all available internal keys within the slot
BackupExternakKeys	Backup all available external keys within the slot
`BackupConfig=`	Backup the slot configuration object
`RestoreInternalKeys=`	Restore all keys from the given key backup file to the internal key store
`RestoreExternakKeys=`	Restore all keys from the given key backup file to the external key store
`RestoreConfig=`	Restore the slot configuration object from the given configuration backup file
`DeleteSO`	Delete the SO
`RecryptExternalKeys`	Creates a backup and recrypt all available external cryptographic keys within the slot with the current MBK

Table 2: p11tool2 commands - overview

2.2 Data Type Notation

The following data type notation is used for the attribute list parameters (`KeyAttr, PubKeyAttr, PrvKeyAttr, CertAttr`) and the attribute template files.

Data Type	Description

`CK_BBOOL`	`CK_TRUE | true | 1`
	`CK_FALSE | false | 0`
	
	Example:
	
	`CKA_PRIVATE=CK_TRUE`

`CK_ULONG`	Unsigned integer value Example: `CKA_MODULUS_BITS=2048`

`RFC2279 String`	String
	
	Example:
	
	`CKA_LABEL=ABC`
	`CKA_LABEL="A B C"`

`Byte Array`	String or hexadecimal notation with "0x" prefix
	
	Example:
	
	`CKA_ID=ABC`
	`CKA_ID=0x414243`
	`CKA_ID="A B C"`

`Big Integer`	Unsigned integer value or hexadecimal notation with "0x" prefix
	
	Example:
	
	`CKA_PUBLIC_EXPONENT=65537`
	`CKA_PUBLIC_EXPONENT=0x010001`

Data Type	Description

`Object Type`	Object type as string
	Example:
	`CKA_CLASS=CKO_PRIVATE_KEY`
	`CKA_KEY_TYPE=CKK_RSA`

`CK_DATE`	Date of format YYYYMMDD Example: `CKA_END_DATE=20141231`

Table 3: Data type notations

3 Basic Commands

These basic commands are p11tool2 internal functions. No connection to the PKCS#11 API will be established.

3.1 Help

If called without any parameter, this command shows a list of all available p11tool2 commands. If a command name is given as a parameter, specific help will be provided.

Syntax	`p11tool2 Help` `p11tool2 Help=<command>`

Parameter	***Description***
`<command>`	specific command of the p11tool2

Example	`csadm Dev=10.17.1.9 CSLGetStatus`

Output	Lists all available p11tool2 commands

3.2 PrintError

This command displays the corresponding error message text to an error code.

Syntax	`p11tool2 PrintError=<err>`

Parameter	***Description***
`<err>`	error code in hexadecimal notation

Example	`p11tool2 PrintError=B901306F`

Output	`Error B901306F`
	`CryptoServer API LINUX`
	`can't get connection`
	`errno = 111`

3.3 Version

This command shows the version number of the p11tool2 and all built-in libraries.

Syntax	`p11tool2 Version`

Output	`p11tool2 3.1.1`
	`p11adm_R2 3.1.1`
	`CryptoServer PKCS#11 Library R3 1.19 (Nov 10 2021)`
	`cxi_api 1.9.0 (Nov 10 2021)`
	`csadm_lib (global) 3.5.1`
	`csapi 1.13.0 (Nov 10 2021)`
	`csxapi 1.11.8 (Nov 10 2021)`
	`pp_api 1.9.8 (Nov 10 2021)`
	`yacl 1.14.4`
	`sdb 2.2.2 (Nov 10 2021 23:32:33)`
	`copa 1.1.8`
	`sl 1.1.5`

4 PKCS#11 Commands

These commands are based on the standard PKCS#11 commands. A connection to the PKCS#11 API will be established.

Note that, by default, the build-in library CryptoServer PKCS#11 Library R3 is used and no shared library is loaded. Thus, the execution of some commands may differ from the standard case. For example, the creation of an SO for token initialization must be authenticated by a user or administrator logged in with a special user type `CKU_CS_GENERIC`. Those special cases are documented in the corresponding command description.

Certain types of shell processes treat certain characters (for example, commas, colons, semi-colons) differently. If the execution of a p11tool2 command fails with an error message from the shell about a missing parameter or an illegal parameter format, quoting parameter values may be necessary.

The following is an example for a correct p11tool2 command syntax in a Microsoft PowerShell:

```
p11tool2 [Slot=<slot_id>] Login="<user_name>,<auth_token>" <command>
```

4.1 Key Usage in FIPS Mode

Each time a DSA/DH/DH_PKCS, RSA or EC (ECDSA or ECDH) key is generated or imported, it must be checked that its usage attribute is exactly one of the { `CKA_SIGN; CKA_VERIFY` } usage bit group or exactly one of the { `CKA_ENCRYPT, CKA_DECRYPT, CKA_DERIVE, CKA_WRAP, CKA_UNWRAP` } usage bit group. I.e., if key pairs are used for signature generation and verification, they must not be used for any other purpose, see *Key Usage in FIPS Mode* and P*adding Mechanisms in FIPS Mode* in the CryptoServer PKCS#11 R3 - Developer Guide.

4.2 ListSlots

This command displays a list of all slots in the system, using the PKCS#11 command `C_GetSlotList`.

Syntax	`p11tool2 ListSlots[=status]`

Parameter	*Description*
`status`	This parameter is optional. If used, the initialization status of each slot is displayed additionally.

Example	`p11tool2 ListSlots`

Output	`0: 00000000`
	`1: 00000001`
	`2: 00000002`
	`3: 00000003`
	`4: 00000004`

4.3 GetInfo

This command displays general information about Cryptoki, using the PKCS#11 command `C_GetInfo`.

Syntax	`p11tool2  GetInfo`

Example	`p11tool2 GetInfo`

Output	CK_INFO:
	
	  cryptokiVersion : 2.20
	
	  manufacturerID 5574696d 61636f20 53616665 77617265
	|Utimaco IS GmbH|
	
	  20414720 20202020 20202020 20202020
	| |
	
	  flags : 0x00000000
	
	  libraryDescription 43727970 746f5365 72766572 20504b43
	|CryptoServer PKC|
	
	  53233131 204c6962 72617279 20523220
	|S#11 Library R3 |
	
	  libraryVersion : 2.13

4.4 GetSlotInfo

This command displays the information about a specific slot, using the PKCS#11 command `C_GetSlotInfo`.

Syntax	`p11tool2  [Slot=<slot_id>]  GetSlotInfo`

Parameter	***Description***
`<slot_id>`	ID of the slot as number. Default: 0

Example	`p11tool2 GetSlotInfo`

Output	`CK_SLOT_INFO (slot ID: 0x00000000):` `  slotDescription  5043493a 30202d20  534c4f54 5f303030 |PCI:0 - SLOT_000|` `  30202020 20202020  20202020 20202020 |0  |` ` 20202020 20202020 20202020 20202020	|` ` 20202020 20202020 20202020 20202020	|`  ` manufacturerID 5574696d 61636f20 53616665 77617265 |Utimaco IS GmbH |` ` 20414720 20202020 20202020 20202020	|`  ` flags: 0x00000005`  ` CKF_TOKEN_PRESENT : CK_TRUE` ` CKF_REMOVABLE_DEVICE : CK_FALSE` ` CKF_HW_SLOT : CK_TRUE`  ` hardwareVersion : 3.00` ` firmwareVersion : 2.01`

4.5 GetTokenInfo

This command displays the information about a specific token, using the PKCS#11 command `C_GetTokenInfo`.

Syntax	`p11tool2  [Slot=<slot_id>]  GetTokenInfo`

Parameter	***Description***
`<slot_id>`	ID of the slot as number. Default: 0

Example	`p11tool2 GetTokenInfo`

Output

```
CK_TOKEN_INFO (slot ID: 0x00000000):

  label               20202020 20202020  20202020 20202020
|                |
                      20202020 20202020  20202020 20202020
|                |

  manufacturerID      5574696d 61636f20  53616665 77617265
|Utimaco IS GmbH |
                      20414720 20202020  20202020 20202020
|                |

  model               43727970 746f5365  72766572 20202020
|CryptoServer    |
  serialNumber        53653530 20202020  43533838 38303232
|Se50    CS888022|

  flags: 0x00000245

    CKF_RNG                             : CK_TRUE
    CKF_WRITE_PROTECTED                 : CK_FALSE
    CKF_LOGIN_REQUIRED                  : CK_TRUE
    CKF_USER_PIN_INITIALIZED            : CK_FALSE
    CKF_RESTORE_KEY_NOT_NEEDED          : CK_FALSE
    CKF_CLOCK_ON_TOKEN                  : CK_TRUE
    CKF_PROTECTED_AUTHENTICATION_PATH   : CK_FALSE
    CKF_DUAL_CRYPTO_OPERATIONS          : CK_TRUE
    CKF_TOKEN_INITIALIZED               : CK_FALSE
    CKF_SECONDARY_AUTHENTICATION        : CK_FALSE
    CKF_USER_PIN_COUNT_LOW              : CK_FALSE
    CKF_USER_PIN_FINAL_TRY              : CK_FALSE
    CKF_USER_PIN_LOCKED                 : CK_FALSE
    CKF_USER_PIN_TO_BE_CHANGED          : CK_FALSE
    CKF_SO_PIN_COUNT_LOW                : CK_FALSE
    CKF_SO_PIN_FINAL_TRY                : CK_FALSE
    CKF_SO_PIN_LOCKED                   : CK_FALSE
    CKF_SO_PIN_TO_BE_CHANGED            : CK_FALSE

  ulMaxSessionCount    : 256
  ulSessionCount       : 0
  ulMaxRwSessionCount  : 256
  ulRwSessionCount     : 0
  ulMaxPinLen          : 255
  ulMinPinLen          : 0
  ulTotalPublicMemory  : -1
  ulFreePublicMemory   : -1
  ulTotalPrivateMemory : -1
  ulFreePrivateMemory  : -1
```

	hardwareVersion : 3.00 firmwareVersion : 2.01 utcTime 32303133 30353033 31353130 33342e30 |20130503151034.0|

4.6 LoginSO

This command logs the security officer (SO) into a token, using the PKCS#11 command C_Login with user type `CKU_SO`.

Syntax	`p11tool2 [Slot=<slot_id>] LoginSO=<so_pin> <command>`

Parameter	***Description***
`<slot_id>`	ID of the slot as number (to open a session). Default: 0
<so_pin>	SO PIN or string 'ask' if hidden PIN entry should be used.
<command>	Command which has to be authenticated by the SO

Example	`p11tool2 LoginSO=654321 InitPIN=123456`

Output	None on success, or error message

4.7 LoginUser

This command logs the normal user into a token, using the PKCS#11 command C_Login with user type `CKU_USER`.

Syntax	`p11tool2 [Slot=<slot_id>] LoginUser=<user_pin> <command>`

Parameter	***Description***
`<slot_id>`	ID of the slot as number (to open a session). Default: 0
`<user_pin>`	User's PIN in clear text or the string 'ask', if hidden PIN entry is preferred.
`<command>`	Command which has to be authenticated by the normal user

Example	`p11tool2 LoginUser=ask GenerateKeyPair=RSA`

Output	None on success, or error message

4.8 Login

This proprietary command only works with the CryptoServer PKCS#11 Library R3.

It logs a user into a token, using the PKCS#11 command `C_Login` with a vendor-defined user type `CKU_CS_GENERIC`.

Syntax	`p11tool2 [Slot=<slot_id>] Login=<user_name>,<auth_token> <command>`

Parameter	*Description*
`<slot_id>`	ID of the slot as number (to open a session). Default: 0
<user_name>	Name of a user
<auth_token>	Authentication token of the user: ▪ Case password-based authentication: User password or string 'ask' if hidden password entry should be used. ▪ Case signature-based authentication: Key specifier where the private part of the user key should be loaded from: • Smartcard specifier, e.g., ':cs2:cjo:USB0' If you append # and a PIN to the smartcard specifier, the user is automatically logged in without any user interaction at the PIN pad. Example: `:cs2:cjo:USB0#123456` See *Authentication Mechanisms* in the *CryptoServer - Administration Manual* for details about authentication mechanisms. • RSA and ECDSA signature authentication The PIN pad has to be connected to the computer where p11tool2 is running (USB port) or to another computer, see the *Using a Local PIN Pad for a Remote CryptoServe*r in the CryptoServer - Administration Manual (p. 99). Example: ' `:cs2:cjo:USB0@123.123.123.123/mypwd` ' • RSA smartcard authentication The smartcard specifier can be omitted. The comma after the user name can be omitted as well (recommended). For an automatic login, replace the smartcard specifier by `#<PIN>` . • Keyfile[#password], e.g., ' `my.key#pwd` ' If the keyfile is encrypted, hidden password entry is possible by entering the string 'ask' as password.

Parameter	*Description*
<command>	Command which has to be authenticated by a user

Example	Example 1: Initialize a token
	`p11tool2 Login=ADMIN,C:/keys/init_prv.key InitToken=654321`
	Example 2:
	The user administrator AdminSc (22000000) uses RSA smartcard authentication and a Utimaco cyberJack one PIN pad directly connected to the CryptoServer PCIe card. He authenticates a `p11tool2 GetGlobalConfig` command to show the global configuration.
	`p11tool2 Login=AdminSc,:cs2:cjo:USB0 GetGlobalConfig=*`
	The following alternatives are possible because `AdminSc` uses RSA smartcard authentication:
	Shorter (no smartcard specifier) but with the same effect:
	`p11tool2 Login= AdminSc, GetGlobalConfig=*`
	Also possible:
	`p11tool2 Login= AdminSc GetGlobalConfig=*`
	With automatic login using the PIN 123456:
	`p11tool2 Login= AdminSc,#123456 GetGlobalConfig=*`

Output	Example 1:
	None on success, or error message
	Example 2:
	`CKA_CFG_ALLOW_SLOTS = CK_TRUE`
	`CKA_CFG_CHECK_VALIDITY_PERIOD = CK_FALSE`
	`CKA_CFG_AUTH_PLAIN_MASK = 0x00000002`
	`CKA_CFG_WRAP_POLICY = CK_FALSE`
	`CKA_CFG_AUTH_KEYM_MASK = 0x00000002`
	`CKA_CFG_SECURE_DERIVATION = CK_FALSE`
	`CKA_CFG_SECURE_IMPORT = CK_FALSE`
	`CKA_CFG_SECURE_RSA_COMPONENTS = CK_TRUE`
	`CKA_CFG_P11R3_BACKWARDS_COMPATIBLE = CK_FALSE`
	`CKA_CFG_ENFORCE_BLINDING = CK_FALSE`
	`CKA_CFG_SECURE_SLOT_BACKUP = CK_FALSE`

4.9 InitToken

This command initializes a token, using the PKCS#11 command `C_InitToken`.

Initialization:

If the token has not been initialized, the given PIN becomes the SO initial PIN.

Note that in case of the CryptoServer PKCS#11 Library R3 a CryptoServer administrator with permission mask 0x20000000 or the default administrator ADMIN must be logged in (via the command Login) to create the SO. The Security Officer (permission 00000200) to be created has the name SO_xxxx with xxxx being a 2-byte decimal representation of the key group/ PKCS#11 slot ID. The name may range from SO_0000 to SO9999.

Reinitialization:

If the token has already been initialized, a reinitialization is only performed if the flag <force> is set to 1 or y. The given SO PIN is used to authorize the reinitialization operation, which will delete all destructible objects and the normal user.

Syntax	`p11tool2 [Slot=<slot_id>] [Label=<label>] [Force=<force>] [Login=<admin_name>,<admin_auth_token>] InitToken=<so_pin>`

Parameter	***Description***
`<slot_id>`	ID of the slot as number. Default: 0
<label>	Label of the token as string or as hexadecimal notation with "0x" prefix. Default: "CryptoServer PKCS11 Token" NOTE: The label is also automatically assigned to the SO of the PKCS#11 slot as the user attribute L[], and is displayed on execution of the csadm command ListUser.
<force>	Boolean flag (0/1 or n/y) to force token re-initialization Default: 0 (= refuse re-initialization)
<admin_name>	Name of a CryptoServer administrator with permission mask 0x20000000 or the default administrator ADMIN

Parameter	*Description*
<admin_auth_token>	Authentication token of the CryptoServer administrator with <admin_name>: ▪ Case password-based authentication: Administrator password or string 'ask' if hidden password entry should be used. ▪ Case signature-based authentication: Key specifier where the private part of the administrator key should be loaded from: • Smartcard specifier, e.g., '`:cs2:cjo:USB0`' See Chapter "Authentication Mechanisms" in [CSADMIN (p. 99)] for details about authentication mechanisms. • RSA and ECDSA signature authentication The PIN pad has to be connected to the computer where p11tool2 is running (USB port) or to another computer (see the Chapter "Using a Local PIN Pad for a Remote CryptoServer" in [CSADMIN (p. 99)]). Example: '`:cs2:cjo:USB0@123.123.123.123/mypwd`' • RSA smartcard authentication The Chapter "Using a Local PIN Pad for a Remote CryptoServer" in [CSADMIN (p. 99)] does not apply here because the PIN pad must be connected directly to the CryptoServer PCIe card. See Chapter "Authentication Mechanisms" in [CSADMIN (p. 99)] for further details. The smartcard specifier can be omitted. The comma after the user name can be omitted as well (recommended). For an automatic login, replace the smartcard specifier by `#<PIN>`. • Keyfile[#password], e.g., '`my.key#pwd`' If the keyfile is encrypted, hidden password entry is possible by entering the string 'ask' as password.
<so_pin>	SO PIN in clear text or the string 'ask' if hidden PIN entry is preferred.

Example	`p11tool2 Slot=2 Login=ADMIN,C:/keys/init_prv.key InitToken=123456` This command creates a Security Officer with the name SO_0002 in the slot SLOT_0002.

Output	none on success, or error message

Verify the result.

Example:

```
csadm Dev=3001@127.0.0.1 ListUser
```

Example output:

```
Name         Permission   Mechanism      Attributes

ADMIN         22000000    RSA sign       Z[0]

SO_0002       00000200    HMAC passwd
A[CXI_GROUP=SLOT_0002]L[CryptoServer PKCS11 Token]
```

If you want to change the label of a token, delete the security officer and initialize a new token with the desired label. Proceed as follows to do so:

1.The token label is shown as the user attribute `L[]` in the output of the csadm ListUser

Example:

```
csadm Dev=3001@127.0.0.1 ListUser
```

Example output:

```
Name         Permission   Mechanism      Attributes

ADMIN         22000000    RSA sign       Z[0]

SO_0002       00000200    HMAC passwd
Z[0]A[CXI_GROUP=SLOT_0002]L[CryptoServer PKCS11 Token       ]
```

2. Delete the security officer of the slot. For details, see Section "DeleteSO (p. 94)".

Example:

```
p11tool2 Slot=2 Login=ADMIN,ADMIN.key DeleteSO
```

3. Initialize a token with the desired label.

Example:

```
p11tool2 Slot=2 Label=MyLabel Login=ADMIN,ADMIN.key InitToken=123456
```

4. Verify the result.

Example:

```
csadm Dev=3001@127.0.0.1 ListUser
```

Example output:

```
Name         Permission    Mechanism      Attributes

ADMIN         22000000     RSA sign       Z[0]

SO_0002       00000200     HMAC passwd
A[CXI_GROUP=SLOT_0002]L[MyLabel       ]
```

4.10 InitPIN

This command initializes the normal user PIN, using the PKCS#11 command C_InitPIN.

The User to be created has the name `USR_xxxx` with `xxxx` being a 2-byte decimal representation of the key group/PKCS#11 slot ID. The name may range from `USR_0000` to `USR_9999`.

Syntax	`p11tool2 [Slot=<slot_id>] LoginSO=<so_pin> InitPIN=<user_pin>`

Parameter	*Description*
<slot_id>	ID of the slot as number (to open a session). Default: 0
<so_pin>	SO PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
<user_pin>	User PIN in clear text or the string 'ask' if hidden PIN entry is preferred.

Example	`p11tool2 Slot=2 LoginSO=123456 InitPIN=123456` This command creates the User with the name `USR_0002` in the slot `SLOT_0002`.

Output	None on success, or error message

Verify the result

Example:

```
csadm Dev=3001@127.0.0.1 ListUser
```

Example output:

```
Name         Permission   Mechanism      Attributes
ADMIN         22000000    RSA sign       Z[0]
SO_0002       0000200    HMAC passwd    Z[0]A[CXI_GROUP=SLOT_0002]L[MyLabel]
USR_0002             00000022             HMAC passwd
 A[CXI_GROUP=SLOT_0002]
```

4.11 SetPIN

This command changes the PIN of the SO or the normal user, using the PKCS#11 command `C_SetPIN`.

Syntax	▪ for SO:
	`p11tool2 [Slot=<slot_id>] LoginSO=<so_pin> SetPIN=<so_pin>,<new_so_pin>`
	▪ for normal user:
	`p11tool2 [Slot=<slot_id>] [LoginUser=<user_pin>] SetPIN=<user_pin>,<new_user_pin>`

Parameter	***Description***
`<slot_id>`	ID of the slot as number (to open a session). Default: 0
`<so_pin>`	(Old) SO PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
`<new_so_pin>`	New SO PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
`<user_pin>`	(Old) user PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
`<new_user_pin>`	New user PIN in clear text or the string 'ask' if hidden PIN entry is preferred.

Example	▪ for SO: `p11tool2 LoginSO=ask SetPIN=ask,ask` ▪ for normal user: `p11tool2 SetPIN=ask,ask`

Output	None on success, or error message

4.12 ListObjects

This command displays a list of available objects, using the PKCS#11 commands like `C_FindObjects` and `C_GetAttributeValue`.

As described in the PKCS#11 standard, private objects are only available with user login. Without user login only public objects are listed.

Syntax	`p11tool2 [Slot=<slot_id>] [LoginUser=<user_pin>] ListObjects`

Parameter	***Description***
`<slot_id>`	ID of the slot as number (to open a session). Default: 0
<user_pin>	User PIN in clear text or the string 'ask' if hidden PIN entry is preferred.

Example	`p11tool2 LoginUser=ask ListObjects`

Output

```
CKO_CERTIFICATE:

+ 1.1

  CKA_CERTIFICATE_TYPE           = CKC_X_509
  CKA_UNIQUE_ID                  = 20F43080-51AE-48C9-
B248-0E8BE2AA63304
  CKA_LABEL                      = P12 Cert
  CKA_ID                         = 0x503132 (P12)
  CKA_SUBJECT                    =

                0x3032310B 30090603 55040613 02444531 |021
0   U    DE1|
                  10300E06 0355040A 13075574 696D6163 |
0   U    Utimac|
                  6F311130 0F060355 04031308 73637465 |o1
0   U    scte|
                  73743031                           |
st01            |
```

```
  CKA_SENSITIVE                   = CK_TRUE
  CKA_EXTRACTABLE                 = CK_FALSE

CKO_PUBLIC_KEY:

+ 2.1

  CKA_KEY_TYPE                    = CKK_RSA
  CKA_UNIQUE_ID                   = 20F43080-51AE-48C9-
B248-0E8BE2AA63304
  CKA_LABEL                       = RSA Public Key
  CKA_ID                          = 0x503132 (P12)
  CKA_SUBJECT                     =

                    0x3032310B 30090603 55040613 02444531 |
021 0   U    DE1|
                    10300E06 0355040A 13075574 696D6163 |
0   U    Utimac|
                    6F311130 0F060355 04031308 73637465 |o1
0   U    scte|

                    73743031                            |
st01            |

  CKA_SENSITIVE                   = CK_TRUE
  CKA_EXTRACTABLE                 = CK_FALSE

+ 2.2

  CKA_KEY_TYPE                    = CKK_ECDSA
  CKA_UNIQUE_ID                   = 20F43080-51AE-48C9-
B248-0E8BE2AA63304
  CKA_LABEL                       = My ECC Public Key
  CKA_ID                          = 0x454343 (ECC)
  CKA_SENSITIVE                   = CK_TRUE
  CKA_EXTRACTABLE                 = CK_FALSE


CKO_PRIVATE_KEY:

+ 3.1

  CKA_KEY_TYPE                    = CKK_RSA
  CKA_UNIQUE_ID                   = 20F43080-51AE-48C9-
B248-0E8BE2AA63304
  CKA_LABEL                       = RSA Private Key
  CKA_ID                          = 0x503132 (P12)
  CKA_SUBJECT                     =

                  0x3032310B 30090603 55040613 02444531 |021
0   U    DE1|
```

```
                    10300E06 0355040A 13075574 696D6163 |
0   U    Utimac|
                    6F311130 0F060355 04031308 73637465 |o1
0   U    scte|
                    73743031                            |
st01            |

  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_TRUE

+ 3.2

  CKA_KEY_TYPE                  = CKK_ECDSA
  CKA_UNIQUE_ID                 = 20F43080-51AE-48C9-
B248-0E8BE2AA63304
  CKA_LABEL                     = My ECC Private Key
  CKA_ID                        = 0x454343 (ECC)
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_TRUE

CKO_SECRET_KEY:

+ 4.1

  CKA_KEY_TYPE                  = CKK_AES
  CKA_UNIQUE_ID                 = 20F43080-51AE-48C9-
B248-0E8BE2AA63304
  CKA_LABEL                     = My AES Secret Key
  CKA_ID                        = 0x414553 (AES)
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
```

For objects that are created using a PKCS#11 provider prior to SecurityServer 4.50 the value of the attribute `CKA_UNIQUE_ID` is displayed as `CK_UNAVAILABLE_INFORMATION`.

Only objects of the following classes are listed:

- `CKO_DATA`
- `CKO_CERTIFICATE`
- `CKO_PUBLIC_KEY`
- `CKO_PRIVATE_KEY`
- `CKO_SECRET_KEY`

- CKO_DOMAIN_PARAMETERS

Only the following object attributes are listed (if set):

- CKA_CERTIFICATE_TYPE
- CKA_KEY_TYPE
- CKA_UNIQUE_ID
- CKA_LABEL
- CKA_ID
- CKA_SUBJECT
- CKA_SENSITIVE
- CKA_EXTRACTABLE

4.13 DeleteObject

This command deletes objects specified by the given label, ID and subject name, using the PKCS#11 commands like `C_FindObjects` and `C_DestroyObject`.

As described in the PKCS#11 specification, private objects can only be deleted with user login. Without user login only public objects can be deleted.

Note that at least one of the label, ID or subject name must be given in order to identify the objects to be deleted.

Syntax	`p11tool2 [Slot=<slot_id>] [LoginUser=<user_pin>] [Label=<label>] [Id=<id>] [Subject=<subject>] DeleteObject`

Parameter	*Description*
`<slot_id>`	ID of the slot as number (to open a session). Default: 0
`<user_pin>`	User PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
`<label>`	Object label as string or as hexadecimal notation with "0x" prefix or '*' for all labels
`<id>`	Object ID as string or as hexadecimal notation with "0x" prefix or '*' for all IDs
`<subject>`	Object subject name as string or as hexadecimal notation with "0x" prefix or '*' for all subject names

Example	`p11tool2 LoginUser=ask Label="RSA Public Key" Id="P12" DeleteObject`

Output	`1 Objects deleted`

4.14 ImportP12

This command imports an X.509 certificate, a public and a private key from the given PKCS#12 file, using the OpenSSL library, and save them as private objects, using the PKCS#11 command `C_CreateObject`.

The object attributes can be overwritten and extended by the attribute list parameters.

Syntax	`p11tool2 [Slot=<slot_id>] LoginUser=<user_pin> [CertAttr=<cert_attr>] [PubKeyAttr=<pub_key_attr>] [PrvKeyAttr=<prv_key_attr>] ImportP12=<filename>,<password>`

Parameter	***Description***
`<slot_id>`	ID of the slot as number (to open a session). Default: 0
`<user_pin>`	User PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
`<cert_attr>`	List of certificate attributes in format `<attribute_name_1>=<value_1>,<attribute_name_2>=<value_2>,…`
`<pub_key_attr>`	List of public key attributes in format `<attribute_name_1>=<value_1>,<attribute_name_2>=<value_2>,…`
`<prv_key_attr>`	Private key attribute list of format `<attribute_name_1>=<value_1>,<attribute_name_2>=<value_2>,…`
`<filename>`	PKCS#12 file
`<password>`	PKCS#12 file password or string 'ask' if hidden password entry is preferred.

Example	`p11tool2 LoginUser=ask CertAttr=CKA_LABEL="P12 Cert",CKA_ID=P12 PubKeyAttr=CKA_LABEL="P12 Public Key",CKA_ID=P12 PrvKeyAttr=CKA_LABEL="P12 Private Key",CKA_ID=0x503132 ImportP12=C:/p12/sctest01.p12,ask`

Output	None on success, or error message

4.14.1 Imported Certified Attributes

The certificate object is created with the following attributes (which can be overwritten or extended by the attribute list `CertAttr`):

Attribute	Value
CKA_CLASS	CKO_CERTIFICATE
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_CERTIFICATE_TYPE	CKC_X_509
CKA_LABEL	"X509 Certificate"
CKA_VALUE	Parsed value from file
CKA_SUBJECT	Parsed subject name from file if set or NULL

Table 4: Attributes of a X.509 certificate

4.14.2 Imported Private Key Attributes

The private key object is created with the following attributes (which can be overwritten or extended by the attribute list `PrvKeyAttr`):

RSA Private Key:

Attribute	Value
CKA_CLASS	CKO_PRIVATE_KEY

Attribute	Value
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_SIGN	CK_TRUE
CKA_EXTRACTABLE	CK_TRUE
CKA_SENSITIVE	CK_TRUE
CKA_KEY_TYPE	CKK_RSA
CKA_LABEL	"RSA Private Key"
CKA_MODULUS	Parsed from file
CKA_PUBLIC_EXPONENT	Parsed from file
CKA_PRIVATE_EXPONENT	Parsed from file
CKA_PRIME_1	Parsed from file
CKA_PRIME_2	Parsed from file
CKA_COEFFICIENT	Parsed from file
CKA_EXPONENT_1	Parsed from file
CKA_EXPONENT_2	Parsed from file
CKA_SUBJECT	Parsed certificate subject name from file if set or NULL

Table 5: Attributes of an RSA private key

DSA Private Key:

Attribute	Value
CKA_CLASS	CKO_PRIVATE_KEY
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_SIGN	CK_TRUE
CKA_EXTRACTABLE	CK_TRUE
CKA_SENSITIVE	CK_TRUE
CKA_KEY_TYPE	CKK_DSA
CKA_LABEL	"DSA Private Key"
CKA_PRIME	Parsed from file
CKA_SUBPRIME	Parsed from file
CKA_BASE	Parsed from file
CKA_VALUE	Parsed from file
CKA_SUBJECT	Parsed certificate subject name from file if set or NULL

Table 6: Attributes of a DSA private key

Private Key:

Attribute	Value
CKA_CLASS	CKO_PRIVATE_KEY

Attribute	Value
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_SIGN	CK_TRUE
CKA_EXTRACTABLE	CK_TRUE
CKA_SENSITIVE	CK_TRUE
CKA_KEY_TYPE	CKK_EC
CKA_LABEL	"ECC Private Key"
CKA_ECDSA_PARAMS	Parsed from file
CKA_VALUE	Parsed from file
CKA_SUBJECT	Parsed certificate subject name from file if set or NULL

Table 7: Attributes of an ECC private key

4.14.3 Imported Public Key Attributes

The public key object is created with the following attributes (which can be overwritten or extended by the attribute list `PubKeyAttr`):

RSA Public Key:

Attribute	*Value*
CKA_CLASS	CKO_PUBLIC_KEY
CKA_TOKEN	CK_TRUE

Attribute	*Value*
CKA_PRIVATE	CK_TRUE
CKA_VERIFY	CK_TRUE
CKA_KEY_TYPE	CKK_RSA
CKA_WRAP	CK_TRUE
CKA_LABEL	"RSA Public Key"
CKA_MODULUS	Parsed from file
CKA_PUBLIC_EXPONENT	Parsed from file
CKA_SUBJECT	Parsed certificate subject name from file if set or NULL

Table 8: Attributes of an RSA public key

DSA Public Key:

Attribute	Value
CKA_CLASS	CKO_PUBLIC_KEY
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_VERIFY	CK_TRUE
CKA_KEY_TYPE	CKK_DSA
CKA_LABEL	"DSA Public Key"
CKA_PRIME	Parsed from file

Attribute	Value
CKA_SUBPRIME	Parsed from file
CKA_BASE	Parsed from file
CKA_VALUE	Parsed from file
CKA_SUBJECT	Parsed certificate subject name from file if set or NULL

Table 9: Attributes of a DSA public key

ECC Public Key:

Attribute	Value
CKA_CLASS	CKO_PUBLIC_KEY
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_VERIFY	CK_TRUE
CKA_KEY_TYPE	CKK_EC
CKA_LABEL	"ECC Public Key"
CKA_ECDSA_PARAMS	Parsed from file
CKA_EC_POINT	Parsed from file

Attribute	Value
CKA_SUBJECT	Parsed certificate subject name from file if set or NULL

Table 10: Attributes of an ECC public key

4.15 ImportCert

This command imports an X.509 certificate and a public key from the given certificate file, using the OpenSSL library, and save them as private objects, using the PKCS#11 command `C_CreateObject`.

The objects' attributes can be overwritten and extended by the attribute list parameters.

Syntax	`p11tool2 [Slot=<slot_id>] LoginUser=<user_pin> [CertAttr=<cert_attr>] [PubKeyAttr=<pub_key_attr>] ImportCert=<filename>`

Parameter	***Description***
<slot_id>	ID of the slot as number (to open a session). Default: 0
<user_pin>	User PIN clear text or the string 'ask' if hidden PIN entry is preferred.

<cert_attr>	List of certificate attributes in format
	`<attribute_name_1>=<value_1>,<attribute_name_2>=`
	`<value_2>,…`

<pub_key_attr>	List of public key attributes in format
	`<attribute_name_1>=<value_1>,<attribute_name_2>=`
	`<value_2>,…`

Parameter	*Description*
`<filename>`	Name of the certificate file to be imported

Example	`p11tool2 LoginUser=ask CertAttr=CKA_LABEL="My Cert",CKA_ID=ABC PubKeyAttr=CKA_LABEL="My Public Key",CKA_ID=0x414243 ImportCert=C:/cert/x509.der`

Output	None on success, or error message

The certificate object is created with the same attributes as for ImportP12.

4.16 ExportCert

This command writes the BER-encoding (attribute CKA_VALUE) of the certificate object which is specified by the given label, ID and subject name to a file or to the standard output if no file name is set. The PKCS#11 commands like `C_FindObjects` and `C_GetAttributeValue` are used.

Note that at least one of the label, ID or subject name must be given in order to identify the certificate object.

Syntax	`p11tool2 [Slot=<slot_id>] LoginUser=<user_pin> [Label=<label>] [Id=<id>] [Subject=<subject>] [Force=<force>] ExportCert[=<filename>]`

Parameter	*Description*
`<slot_id>`	ID of the slot as number (to open a session). Default: 0

Parameter	*Description*
<user_pin>	User PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
<label>	Label of the certificate object as string or as hexadecimal notation with "0x" prefix
<id>	ID of the certificate object as string or as hexadecimal notation with "0x" prefix
<subject>	certificate subject name as string or as hexadecimal notation with "0x" prefix
<force>	Boolean flag (0/1 or n/y) to overwrite file if already exists. Default: 0 (Cancel command if file already exists)
<filename>	Output file (default: standard output)

Example	`p11tool2 LoginUser=ask Label="P12 Cert" Id="P12" ExportCert`

Output	`Certificate value:`
	`  CKA_VALUE  =`
	`  0x3082026C 308201D5 A0030201 02020106 |0 l0  |`
	`  300D0609 2A864886 F70D0101 05050030 |0 * H  0|`
	`  2E310D30 0B060355 04031304 726F6F74 |.1 0  U  root|`
	`  310B3009 06035504 06130244 45311030 |1 0  U  DE1 0|`
	`  0E060355 040A1307 5574696D 61636F30 | U  Utimaco0|`
	`  1E170D30 34303130 31313230 3030305A | 040101120000Z|`

```
                    170D3037 30313031 31323030 30305A30 |
07010112000OZ0|

                    32310B30 09060355 04061302 44453110 |21
0   U    DE1 |

                    300E0603 55040A13 07557469 6D61636F |0
U    Utimaco|

                    3111300F 06035504 03130873 63746573 |1
0   U    sctes|

                    74303130 819F300D 06092A86 4886F70D |
t010  0   * H   |

                    01010105 0003818D 00308189 02818100 |
         0      |

                    B8AC59FF 544BF8EA 4791300A 70B9420C |  Y
TK  G 0 p B |

                    648DAA23 0BB1A5BA 71C8D5FD 094F728F |d
#    q    Or |

                    F740393B 3BF0752D 16D06C5B 57E83555 |
@9;; u-  l[W 5U|

                    E40EBB63 7B8BCC6B 6ADDD9F7 78A56AF5 |
c{  kj   x j |

                    43AFB193 ED5E40E0 6E663A82 E5BB6FA3 |
C    ^@ nf:   o |

                    8D933445 15932465 C8977CAF E6865ED0 |
4E  $e  |   ^ |

                    FE822B7D 8A287761 3F110EFF A9FAAF8E |
+} (wa?       |

                    3A293EA3 DC890996 5BA830FF 27BB350B
|:)>     [ 0 ' 5 |

                    02030100 01A38195 30819230 09060355 |
        0  0   U|

                    1D130402 3000300E 0603551D 0F0101FF |
0 0   U     |

                    04040302 04B0301D 0603551D 0E041604 |
      0   U     |
```

```
                    14919DFD 50486F78 0FB51520 7C1CDCEC |
PHox    |   |

                    9B1F19FF 86305606 03551D23 044F304D |
     0V  U # O0M|

                    80141309 CE05C9CE 632381DB B8DB65D1 |
        c#    e |

                    EFABAA84 34FFA132 A430302E 310D300B |
4  2 00.1 0 |

                    06035504 03130472 6F6F7431 0B300906 |
U    root1 0  |

                    03550406 13024445 3110300E 06035504 |
U    DE1 0   U |

                    0A130755 74696D61 636F8201 01300D06 |
Utimaco   0  |

                    092A8648 86F70D01 01050500 03818100 | *
H            |

                    66352161 049026CE 31E26F7A B2BA1B3E |f5!
a  & 1 oz   >|

                    DD03E263 0879371A 91E6FF89 D5DD9316 |
c y7         |

                    8FCF4C98 B025B98D 7DACF7C8 66C0F73E |
L  %  }   f  >|

                    25947245 9FF451BA C0B729B7 D9B88B94 |%
rE  Q   )     |

                    FAF133B0 208A5FB9 BBFB7382 B8B209A0 |
3   _   s     |

                    B7C94ED3 62624BBC 7C6CEA84 337071D3 |  N
bbK |l  3pq |

                    418025A1 19D0E90E 50A034E7 D00E76AD |A
%     P 4   v |

                    B12A22EA 9E309CEE 3294CEA6 05CABF0A |
*"  0  2       |

                    F7E4E701 00000000 03C70040 01000000 |
           @    |

                    B8FD1200 00000000 F7E4E701 00000000 |
                |
```

```
                  01000000 00000000 3045E801 00000000 |
        0E        |
                  FEFFFFFF FFFFFFFF B07BE701 00000000 |
         {        |
                  00000000 00000000 8094E701 00000000 |
                  |
                  7079E701 00000000 01000000 00000000 |
py                |
                  80961C40 01000000 00000000 00000000 |
@             |
                  B8FD1200 00000000 F09EE701 00000000 |
                  |
                  B4AB6B01 00000000 5D8D0340 01000000 |
k     ]  @    |
                  00000000 00000000 02000000 00000000 |
                  |
                  F7E4E701 00000000 F7E4E701 F7E4E701 |
                  |
                  009BE701 00000000 009BE701 00000000 |
                  |
                  10FE1200 00000000 E09AE701 00000000 |
                  |
                  FEFFFFFF FFFFFFFF 68FE1200 00000000 |
        h         |
                  00000000 00000000 00000000 00000000 |
                  |
                  00000000 00000000 0F000000 00000000 |
                  |
                  0F000000 00000000 0043493A 30000000 |
         CI:0     |
                  38010000 00000000 00000000 00000000 |
8                 |
                  68FE1200 00000000 A7A80440 01000000 |
h           @     |
```

```
                    01000000 00000000 0079E701 00000000 |
          y       |
                    00000000 00000000 00000000 00000000 |
                  |
                    0F000000 00000000 8DE70040 01000000 |
            @     |
                    006C6F74 00000000 F06CE701 00000000 |
lot     l       |
                    00000000 00000000 0F000000 00000000 |
                  |
                    FEFFFFFF FFFFFFFF F0F45174 00000000 |
           Qt     |
```

4.17 ExportP10

`This command exports a certificate request for a cryptographic key in ASN.1 encoded PKCS#10 format into a CSR file (certificate signing request).`

Syntax	`p11tool2 [Slot=<slot_id>] LoginUser=<user_pin> [PubKeyAttr=<pub_key_attr>] [PrvKeyAttr=<prv_key_attr>] [Mech=<mech>] [Config=<cfgfile>] [DN=<dn>] [Force=<force>] ExportP10=<filename>`

Parameter	***Description***
`<slot_id>`	ID of the slot as number (to open a session). Default: `0`
`<user_pin>`	User PIN in clear text or the string `ask` if hidden PIN entry is preferred.

Parameter	*Description*
<pub_key_attr>	List of public key attributes in format `<attribute_name_1>=<value_1>,<attribute_name_2>=<value_2>,…`
<prv_key_attr>	List of private key attributes in format `<attribute_name_1>=<value_1>,<attribute_name_2>=<value_2>,…`
<mech>	Mechanism

Parameter	*Description*
<cfgfile>	The absolute path and/or name of the distinguished name configuration file. If no path is specified, the file is searched in the current directory. Either the `Config` parameter or the `DN` parameter must be available. The following distinguished name fields are supported: ▪ `commonName` Specifies an identifier of an object. ▪ `serialNumber` Serial number of the distinguished name. Must be a hexadecimal value without a leading " `0x` " but with up to 40 digits. ▪ `countryName` The ISO code consisting of two capital letters for the country where the organization is located, for example, `GB` , `FR` or `US` ▪ `localityName` Town or city ▪ `stateOrProvinceName` Province, region, county or state, for example, `Sussex` , `Normandy` or `New Jersey` . Do not use abbreviations here. ▪ `organizationName` Company name including suffixes such as `Inc.` or `Corp.` ▪ `organizationalUnitName` Department name, for example, `HR` , `Finance` or `IT` ▪ `title` ▪ `description`

Parameter	*Description*

	▪ `emailAddress`
	An email address to contact the organization
	
	The order of the fields is irrelevant. At least one field must be available. Each field may be available several times.
	
	Example 1 of the contents of the configuration file:
	`commonName=1234_Key25`
	`serialNumber=123`
	`countryName=DE`
	`localityName=Aachen`
	`stateOrProvinceName=Rhineland`
	`organizationName=Utimaco IS GmbH`
	`organizationalUnitName=Support`
	`title=Support`
	`description=Test`
	`emailAddress=support@utimaco.com`
	
	Example 2:
	`organizationName=Utimaco IS GmbH`
	`CommonName=test`
	
	The following short distinguished names can be used:
	`CN: CommonName`
	`L: localityName`
	`C: countryName`
	`ST: stateOrProvinceName`
	`O: organizationName`
	`OU: organizationalUnitName`
	
	Example 3:
	`O=Utimaco IS GmbH`
	`CN=test`

Parameter	*Description*

<dn>	Sequence of distinguished name fields separated by a comma.
	Either the `Config` parameter or the `DN` parameter must be available.
	The description of the `<cfgfile>` parameter applies to the `<dn>` parameter in an analog way.
	If a distinguished name field value contains a blank, the complete DN value must be set in quotation marks.
	Example 1:
	`DN="organizationName=Utimaco IS GmbH,CommonName=test"`
	Example 2 (with short distinguished names):
	`DN="O=Utimaco IS GmbH,CN=test"`

<force>	Boolean flag (`0` / `1` or `n` / `y`) to overwrite file if already exists. Default: `0` (Cancel command if file already exists)
<filename>	The absolute path and/or name of the certificate request file to be generated. The `<force>` parameter decides whether a file with this name is overwritten. If a path is specified, a filename must be specified as well. If no path is specified, the file is created in the current directory.

Example	`p11tool2 LoginUser=123456 PubKeyAttr=CKA_LABEL="My RSA Public Key",`
	`CKA_ID=0x525341 PrvKeyAttr=CKA_LABEL="My RSA Private Key",`
	`CKA_ID=RSA Mech=CKM_SHA256_RSA_PKCS_PSS Config=my.cfg`
	`ExportP10=test.csr`

Output	Upon successful execution of the command, no output is given.
	
	Example for the output if the output file already exists:
	`Error: Command canceled. File test.csr already exists.`
	`Set parameter Force=1 (or y) in front of the command to overwrite this file.`

`The result can be verified by using the openssl command.`

`Verifying the example:`

```
openssl req -text -inform der -in test.csr | grep "Subject:"
```

4.18 GenerateKeyPair

This command generates a public/private key pair, using the PKCS#11 command `C_GenerateKeyPair`.

The key pair can be generated in two different ways:

- Key pair based on default attribute template
- Key pair from template file

4.18.1 Generate Key Pair Based on Default Attribute Template

A key pair with the given mechanism will be generated using default templates which can be overwritten and extended by the attribute list parameters.

Syntax	`p11tool2 [Slot=<slot_id>] LoginUser=<user_pin> [PubKeyAttr=<pub_key_attr>]`
	
	`[PrvKeyAttr=<prv_key_attr>] GenerateKeyPair=<mech>`

Parameter	*Description*
<slot_id>	ID of the slot as number (to open a session). Default: 0
<user_pin>	User PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
<pub_key_attr>	List of public key attributes in format <attribute_name_1>=<value_1>,<attribute_name_2>=<value_2>,…
<prv_key_attr>	List of private key attributes in format <attribute_name_1>=<value_1>,<attribute_name_2>=<value_2>,…
<mech>	Mechanism type: RSA | ECC

Example	p11tool2 LoginUser=ask PubKeyAttr=CKA_LABEL="My RSA Public Key",CKA_ID=0x525341 PrvKeyAttr=CKA_LABEL="My RSA Private Key",CKA_ID=RSA GenerateKeyPair=RSA p11tool2 Slot=1 LoginUser =212223 PubKeyAttr=CKA_LABEL="My ECC Public Key",CKA_ID=0x454343303032,CKA_EC_PARAMS="secp256r1" PrvKeyAttr=CKA_LABEL="new ECC priv002",CKA_ID=ECC generatekeypair=ECC

Output	None on success, or error message

Default RSA Public Key Template:

Attribute	*Value*
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE

Attribute	*Value*
CKA_VERIFY	CK_TRUE
CKA_ENCRYPT	CK_TRUE
CKA_WRAP	CK_TRUE
CKA_MODULUS_BITS	2048
CKA_PUBLIC_EXPONENT	0x010001
CKA_LABEL	"RSA Public Key"

Table 11: Default attribute values for an RSA public key

Default RSA Private Key Template:

Attribute	Value
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_SENSITIVE	CK_TRUE
CKA_EXTRACTABLE	CK_TRUE
CKA_SIGN	CK_TRUE
CKA_DECRYPT	CK_TRUE
CKA_UNWRAP	CK_TRUE
CKA_LABEL	"RSA Private Key"

Table 12: Default attribute values for an RSA private key

Default ECC Public Key Template:

Attribute	Value
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_VERIFY	CK_TRUE
CKA_EC_PARAMS	secp256r1
CKA_LABEL	"ECC Public Key"

Table 13: Default attribute values for an ECC public key

Default ECC Private Key Template:

Attribute	Value
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_SENSITIVE	CK_TRUE
CKA_EXTRACTABLE	CK_TRUE
CKA_SIGN	CK_TRUE
CKA_LABEL	"ECC Private Key"

Table 14: Default attribute values for an ECC private key

4.18.2 Generate Key Pair from Template File

A key pair will be generated using the given template file.

Syntax	`p11tool2 [Slot=<slot_id>] LoginUser=<user_pin> GenerateKeyPair=<template_file>`

Parameter	***Description***
`<slot_id>`	ID of the slot as number (to open a session). Default:
`<user_pin>`	User PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
`<template_file>`	Template file with sections [Mechanism], [PublicKey] and [PrivateKey]

Example	`p11tool2 LoginUser=ask GenerateKeyPair=C:/rsa_keypair_template.txt`

Output	None on success, or error message

Template File:

The template file must contain the following sections:

Section	***Description***
`[Mechanism]`	Contains only one variable: CK_MECHANISM_TYPE Example: `CK_MECHANISM_TYPE = CKM_RSA_PKCS_KEY_PAIR_GEN`

Section	*Description*
[PublicKey]	Contains public key attributes Example: CKA_TOKEN = CK_TRUE CKA_LABEL = "RSA Public Key" …
[PrivateKey]	Contains private key attributes Example: CKA_TOKEN = CK_TRUE CKA_LABEL = "RSA Private Key" …

Table 15: Sections of a template file for key pair generation

Example:

```
[Mechanism]
CK_MECHANISM_TYPE = CKM_RSA_PKCS_KEY_PAIR_GEN

[PublicKey]
CKA_TOKEN = CK_TRUE
CKA_PRIVATE = CK_TRUE
CKA_ENCRYPT = CK_TRUE
CKA_VERIFY = CK_TRUE
CKA_WRAP = CK_TRUE
CKA_MODULUS_BITS = 2048
CKA_PUBLIC_EXPONENT = 0x010001
CKA_LABEL = "My RSA Public Key"
```

```
CKA_ID = 0x525341


[PrivateKey]
CKA_TOKEN = CK_TRUE
CKA_PRIVATE = CK_TRUE
CKA_SENSITIVE = CK_TRUE
CKA_DECRYPT = CK_TRUE
CKA_EXTRACTABLE = CK_TRUE
CKA_SIGN = CK_TRUE
CKA_UNWRAP = CK_TRUE
CKA_LABEL = "My RSA Private Key"
CKA_ID = RSA
```

4.19 GenerateKey

This command generates a secret key or set of domain parameters, using the PKCS#11 command C_GenerateKey.

The secret key can be generated in two different ways:

- Secret key based on default attribute template
- Secret key or domain parameters from template file.

4.19.1 Generate Secret Key Based on Default Attribute Template

A key pair with the given mechanism will be generated using default templates which can be overwritten and extended by the attribute list parameters.

Syntax	`p11tool2 [Slot=<slot_id>] LoginUser=<user_pin> [KeyAttr=<key_attr>] GenerateKey=<mech>`

Parameter	*Description*
<slot_id>	ID of the slot as number (to open a session). Default: 0
<user_pin>	User PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
<key_attr>	Secret key attribute list of format <attribute_name_1>=<value_1>,<attribute_name_2>=<value_2>, …
<mech>	Mechanism type: AES | DES | DES2 | DES3

Example	`p11tool2 LoginUser=ask KeyAttr=CKA_LABEL="My AES Secret Key",CKA_ID=0x414553 GenerateKey=AES`

Output	None on success, or error message

Default AES Key Template

Attribute	*Value*
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_DECRYPT	CK_TRUE
CKA_SIGN	CK_TRUE
CKA_ENCRYPT	CK_TRUE
CKA_WRAP	2048

Attribute	*Value*
CKA_LABEL	"AES Secret Key"
CKA_VALUE_LEN	32

Table 16: Default attribute values for an AES key

Default DES Key Template

Attribute	*Value*
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_DECRYPT	CK_TRUE
CKA_SIGN	CK_TRUE
CKA_ENCRYPT	CK_TRUE
CKA_WRAP	CK_TRUE
CKA_LABEL	"DES Secret Key"

Table 17: Default attribute values for an DES key

Default DES 2 Key Template

Attribute	*Value*
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE

Attribute	*Value*
CKA_DECRYPT	CK_TRUE
CKA_SIGN	CK_TRUE
CKA_ENCRYPT	CK_TRUE
CKA_WRAP	CK_TRUE
CKA_LABEL	"DES2 Secret Key"

Table 18: Default attribute values for an DES2 key

Default DES 2 Key Template

Attribute	*Value*
CKA_TOKEN	CK_TRUE
CKA_PRIVATE	CK_TRUE
CKA_DECRYPT	CK_TRUE
CKA_SIGN	CK_TRUE
CKA_ENCRYPT	CK_TRUE
CKA_WRAP	CK_TRUE
CKA_LABEL	"DES3 Secret Key"

Table 19: Default attribute values for an DES3 key

4.19.2 Generate Secret Key from Template File

A secret key or set of domain parameters will be generated using the given template file.

Syntax	`p11tool2 [Slot=<slot_id>] LoginUser=<user_pin> GenerateKey=<template_file>`

Parameter	***Description***
`<slot_id>`	ID of the slot as number (to open a session). Default: 0
`<user_pin>`	User PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
`<template_file>`	Template file with sections [Mechanism] and [Key]

Example	`p11tool2 LoginUser=123456 GenerateKey=C:/aes_key_template.txt`

Output	None on success, or error message

Template File

The template file must contain the following sections:

Section	***Description***
`[Mechanism]`	Contains only one variable: CK_MECHANISM_TYPE Example: `CK_MECHANISM_TYPE = CKM_AES_KEY_GEN`

Section	*Description*

[Key]	Contains key attributes
	Example:
	`CKA_CLASS=CKO_SECRET_KEY`
	`CKA_KEY_TYPE = CK_DES2CKK_AES`
	`CKA_LABEL = "RSA Public Key"`

Table 20: Sections of a template file for the generation of secret key

Example

```
[Mechanism]
CK_MECHANISM_TYPE = CKM_AES_KEY_GEN
[Key] CKA_CLASS = CKO_SECRET_KEY
CKA_KEY_TYPE = CKK_AES
CKA_TOKEN = CK_TRUE
CKA_PRIVATE = CK_TRUE
CKA_DECRYPT = CK_TRUE
CKA_SIGN = CK_TRUE
CKA_ENCRYPT = CK_TRUE
CKA_VERIFY = CK_TRUE
CKA_WRAP = CK_TRUE
CKA_VALUE_LEN = 32
CKA_LABEL = "My AES Secret Key"
CKA_ID = 0x414553
```

5 Configuration Commands

These configuration commands are proprietary and not part of PKCS#11. They only work with the CryptoServer PKCS#11 Library R3 which has vendor defined extensions.

The CryptoServer PKCS#11 Library R3 provides special PKCS#11 objects called the configuration objects:

- Local Configuration Object - used for configurations that affect the instance of the PKCS#11 API
- Global CryptoServer Configuration Object - used for configurations that affect the whole CryptoServer. These configuration objects can be modified by using the p11tool2 configuration command `SetGlobalConfig`.
- Slot CryptoServer Configuration Object - Used for configurations that affect the current slot. These configuration objects can be modified using the p11tool2 configuration command `SetSlotConfig`.

Changes on the attributes of the global and slot configuration objects are deleted on alarm occurrence and when the Clear command (see Chapter "The Clear Functionality" in [CSADMIN (p. 99)]) is performed. Therefore, we highly recommend to create a backup of the configuration objects:

- To back up the Slot CryptoServer Configuration Object, use the p11tool2 command `BackupConfig`.

To back up the Global CryptoServer Configuration Object, use the csadm command BackupDatabase as described in Chapter "BackupDatabase" of the [CSADMIN (p. 99)].

5.1 ListConfig

This command displays a list of all configuration attributes, see SetGlobalConfig (p. 70).

Syntax	`p11tool2 ListConfig`

Example	`p11tool2 ListConfig`

Output	a) Local Configuration Object: Supported attributes type: CKA_UTIMACO_CFG_PATH <string> b) Global CryptoServer Configuration Object: Supported attributes type: CKA_CFG_ALLOW_SLOTS <bool> CKA_CFG_CHECK_VALIDITY_PERIOD <bool> CKA_CFG_AUTH_PLAIN_MASK <unsigned integer (hex)> CKA_CFG_WRAP_POLICY <bool> CKA_CFG_AUTH_KEYM_MASK <unsigned integer (hex)> CKA_CFG_SECURE_DERIVATION <bool> CKA_CFG_SECURE_IMPORT <bool> CKA_CFG_SECURE_RSA_COMPONENTS <bool> CKA_CFG_P11R3_BACKWARDS_COMPATIBLE <bool> CKA_CFG_ENFORCE_BLINDING <bool> CKA_CFG_SECURE_SLOT_BACKUP <bool> CKA_CFG_ENFORCE_EXT_KEYS <bool> CKA_CFG_ALLOW_WEAK_DES_KEYS <bool> c) Slot CryptoServer Configuration Object: Supported attributes type: CKA_CFG_CHECK_VALIDITY_PERIOD <bool>

```
    CKA_CFG_AUTH_PLAIN_MASK             <unsigned integer
(hex)>

    CKA_CFG_WRAP_POLICY                 <bool>

    CKA_CFG_AUTH_KEYM_MASK              <unsigned integer
(hex)>

    CKA_CFG_SECURE_DERIVATION           <bool>

    CKA_CFG_SECURE_IMPORT               <bool>

    CKA_CFG_SECURE_RSA_COMPONENTS       <bool>

    CKA_CFG_P11R3_BACKWARDS_COMPATIBLE  <bool>

    CKA_CFG_ENFORCE_BLINDING            <bool>

    CKA_CFG_SECURE_SLOT_BACKUP          <bool>

    CKA_CFG_SLOT_BACKUP_PASS_HASH       <string>

    CKA_CFG_ENFORCE_EXT_KEYS            <bool>

    CKA_CFG_ALLOW_WEAK_DES_KEYS         <bool>
```

`CKA_CFG_ENFORCE_EXT_KEYS` and `CKA_CFG_ALLOW_WEAK_DES_KEYS` are only available as of SecurityServer 4.30 (CXI firmware module version 2.4.0.0 and later).

5.2 GetLocalConfig

This command displays the value of the local configuration object attribute with the given name.

Syntax	`p11tool2  GetLocalConfig=<attribute>`

Parameter	*Description*
<attribute>	Name of the local configuration attribute or '*' to get all local configuration attribute values

Example	p11tool2 GetLocalConfig=CKA_UTIMACO_CFG_PATH

Output	CKA_UTIMACO_CFG_PATH = C: \ProgramData\Utimaco\PKCS11_R3\cs_pkcs11_R3.cfg

5.3 GetGlobalConfig

This command displays the value of the global configuration object attribute with the given name.

Syntax	p11tool2 [Slot=<slot_id>] <login_command> GetGlobalConfig=<attribute>

Parameter	*Description*
<slot_id>	ID of the slot as number (to open a session). Default: 0
<login_command>	▪ Login as SO (via LoginSO) or ▪ Login as normal user (via LoginUser) or ▪ Login as administrator, key manager or key user (via Login)

Parameter	*Description*
<attribute>	Name of the global configuration attribute or '*' to get all global configuration attribute values; See SetGlobalConfig for the list of available attributes and values.

Example	`p11tool2 LoginUser=ask GetGlobalConfig=CKA_CFG_ALLOW_SLOTS`

Output	`CKA_CFG_ALLOW_SLOTS = CK_FALSE`

5.4 SetGlobalConfig

This command sets the value of the global configuration object attribute with the given name to the given value.

Syntax	`p11tool2 [Slot=<slot_id>] Login=<admin_name>,<admin_auth_token> SetGlobalConfig=<attribute>,<value>`

Parameter	*Description*
<slot_id>	ID of the slot as number (to open a session). Default: 0
<admin_name>	Name of a CryptoServer administrator with permission mask 0x20000000

Parameter	*Description*
<admin_auth_token>	Authentication token of the CryptoServer administrator with <admin_name>: ▪ Case password-based authentication: Administrator password or string 'ask' if hidden password entry should be used. ▪ Case signature-based authentication: Key specifier where the private part of the administrator key should be loaded from: • Smartcard specifier, e.g., ':cs2:cjo:USB0' See Chapter "Authentication Mechanisms" in [CSADMIN (p. 99)] or [CSMSADM (p. 99)] for details about authentication mechanisms. • RSA and ECDSA signature authentication The PIN pad has to be connected to the computer where p11tool2 is running (USB port) or to another computer (see the Chapter "Using a Local PIN Pad for a Remote CryptoServer" in [CSADMIN (p. 99)]). Example: ':cs2:cjo:USB0@123.123.123.123/mypwd' • RSA smartcard authentication The Chapter "Using a Local PIN Pad for a Remote CryptoServer" in [CSADMIN (p. 99)] does not apply here because the PIN pad must be connected directly to the CryptoServer PCIe card. See Chapter "Authentication Mechanisms" in [CSADMIN (p. 99)] or [CSMSADM (p. 99)] for further details. The smartcard specifier can be omitted. The comma after the user name can be omitted as well (recommended). For an automatic login, replace the smartcard specifier by #<PIN>. • Keyfile[#password], e.g., 'my.key#pwd' If the keyfile is encrypted, hidden password entry is possible by entering string 'ask' as password.
<attribute>	Name of the global configuration attribute; see table below for the list of available attributes and values.
<value>	Value of the global configuration attribute to be set; see table below for the list of available global configuration attributes and possible values.

Example	`p11tool2 Login=ADMIN,C:/keys/init_prv.key SetGlobalConfig=CKA_CFG_ALLOW_SLOTS,CK_TRUE`

Output	none on success, or error message

The following table provides a list of all available configuration objects and their possible values.

Attribute	Description
CKA_CFG_ALLOW_SLOTS	This attribute enables the Security Officer (SO) to configure slots. Type: CK_BBOOL ▪ CK_TRUE - the Security Officer is permitted to configure slots. ▪ CK_FALSE (default) - the Security Officer (SO) is not permitted to configure slots.
CKA_CFG_CHECK_VALIDITY_PERIOD	This attribute checks the validity period of the key. Type: CK_BBOOL The validity period of a key is only checked, if the following functions are to be performed using the key: C_SignInit (), C_EncryptInit (), C_DecryptInit (), C_DeriveInit (), C_WrapKey (), C_UnwrapKey () Possible values: ▪ CK_TRUE - the validity period of a key is checked, if the key has the attributes CKA_START_DATE and CKA_END_DATE. ▪ CK_FALSE (default) - the validity period of a key is not checked.

Attribute	Description
CKA_CFG_AUTH_PLAIN_MASK	This attribute defines the permissions required to import and export a key in plaintext. Type: CK_ULONG Default value: 0x00000002 - corresponds to the permissions of the Cryptographic User, who is already set up in the CryptoServer. **IMPORTANT:** If you change the default setting, you must also use the csadm administration tool to set up the corresponding user in your CryptoServer. This user must be assigned the permissions specified here. Examples for creating different users with csadm are provided in Chapter "AddUser" of the [CSADMIN (p. 99)].
CKA_CFG_WRAP_POLICY	This attribute applies a key wrapping policy specifying how keys are encrypted so they can be securely exported outside the CryptoServer. Type: CK_BBOOL Possible values: ▪ CK_TRUE - a strong key (for example, 256-bit AES) cannot be encrypted with a weak key (for example, 1024-bit RSA). ▪ CK_FALSE (default) - a strong key can be encrypted with a weak key.

Attribute	Description
CKA_CFG_AUTH_KEYM_MASK	This attribute defines the min. required authentication status of the key manager who, by default, has the same permissions as the User (0x00000002). Type: CK_ULONG Default Value: 0x00000002 Allowed values: 0x00000002(default), 0x00000020 IMPORTANT: If the User role has been split into the key user role (0x00000002) and the key manager role (0x00000020), the steps in Chapter "Separated Key Manager and Key User Role (p. 84)" must be performed.
IMPORTANT NOTE: The following security relevant attributes are available as from SecurityServer 4.01 (CXI firmware module version 2.1.11.1).	

Attribute	Description
CKA_CFG_SECURE_DERIVATION	This attribute prohibits the use of the following key derivation mechanisms, and prevents Reduced Key Space attacks: ▪ CKM_XOR_BASE_AND_DATA ▪ CKM_CONCATENATE_DATA_AND_BASE ▪ CKM_CONCATENATE_BASE_AND_DATA ▪ CKM_CONCATENATE_BASE_AND_KEY ▪ CKM_EXTRACT_KEY_FROM_KEY For a detailed description of the mechanisms see [PKCS11ICMS (p. 99)]. Type: CK_BBOOL Possible values: ▪ CK_TRUE – none of the key derivation mechanisms listed above can be used by the function C_Derive (). ▪ CK_FALSE (default) – the key derivation mechanisms listed above can be used by the function C_Derive () for key derivation.
CKA_CFG_SECURE_IMPORT	This attribute prevents simple Key Extraction attacks by performing additional strict checks on wrapping keys. Type: CK_BBOOL Possible values: ▪ CK_TRUE – the key wrapping and unwrapping functions perform additional strict checks on wrapping keys. For more details about the additional checks, please read Chapter "Global CryptoServer Configuration Object" in [CS_PKCS11DEV (p. 99)]. ▪ CK_FALSE (default) – no additional strict checks on wrapping keys are performed.

Attribute	Description
CKA_CFG_SECURE_RSA_COMPONENTS	This attribute applies restrictions on the length of the public exponent used for the generation of RSA keys. Type: CK_BBOOL Possible values: ▪ CK_TRUE (default) – new RSA keys cannot be created with very low, smaller than 0x10001, public exponents. ▪ CK_FALSE – new RSA keys can be created with very low public exponents.
CKA_CFG_P11R3_BACKWARDS_COMPATIBLE	This attribute determines whether keys can be used by default as base keys for key derivation or not. Type: CK_BBOOL ▪ CK_TRUE – keys generated by using an ECC scheme or Diffie-Hellman algorithm can be used as base keys for key derivation (PKCS#11 standard non-compliant legacy); may be necessary for some integrations. ▪ CK_FALSE (default) –newly generated or imported keys cannot be used by default as base keys for key derivation.

Attribute	Description
CKA_CFG_ENFORCE_BLINDING	This attribute prevents side-channel analysis (SCA) attacks by enabling/disabling CryptoServer-specific software measures for SCA resistance. These software measures imply changing the internal computations of RSA and ECC keys in a way that Simple and Differential Power Analysis (also referred to as SPA and DPA), Electro Magnetic Analysis (EMA) and Timing Analysis (TA) measurements on cryptographic keys do not reveal information any longer. However, the measures for SCA resistance negatively affect the performance of the cryptographic operations on RSA and ECDSA keys. Therefore, they are disabled by default, and can be enabled, if necessary. Type: CK_BBOOL ▪ CK_TRUE – software measures for SCA resistance are used for cryptographic operations on RSA and ECDSA keys. ▪ CK_FALSE (default) – normal (without software measures for SCA resistance) cryptographic operations on RSA and ECDSA keys are used.
IMPORTANT NOTE: The following security relevant attribute is available as from SecurityServer 4.10 (CXI firmware module version 2.2.1.0 and later).	

Attribute	Description
CKA_CFG_SECURE_SLOT_BACKUP	This attribute enforces the usage of an individual backup key (Tenant Backup Key, TBK) per slot instead of the MBK to protect external keys and key backups. By default, only MBK-protected external key storage and key backup is enabled. Possible values: ▪ CK_TRUE – use slot-individual backup keys (TBKs) derived from the CryptoServer's MBK to encrypt external keys and key backups.

Attribute	Description
	▪ CK_FALSE (default) – use the CryptoServer's MBK to encrypt external keys and key backups. Make sure you have set this configuration attribute according to your security policy before your CryptoServer production environment gets operational. It is optional, but recommended, to use a passphrase for the derivation of a slot-individual backup key. This is done by setting the CKA_CFG_SLOT_BACKUP_HASH_PASS configuration attribute. VERY IMPORTANT: If you use the CKA_CFG_SLOT_BACKUP_HASH_PASS configuration attribute, make sure you have set it before you set the CKA_CFG_SECURE_SLOT_BACKUP configuration attribute. See Chapter 5.7, “SecureSlotPass”, for details. If you use SecurityServer/CryptoServer SDK 4.10, create an external key or key backup by using an MBK, then enable the usage of slot-individual backup keys by setting the CKA_CFG_SECURE_SLOT_BACKUP configuration attribute to the CK_TRUE value, trying to restore the external key or key backup fails and the external key and key backups become inaccessible. The error message “invalid mac of key blob” (error code: 0xB0680026) is created. This applies as well if you have upgraded to SecurityServer/ CryptoServer SDK 4.20 or later before trying to restore the external key or key backups. However, if you upgrade to SecurityServer/CryptoServer SDK 4.20 before creating the external key or key backup using an MBK, restoring them with a slot-individual backup key will succeed.
IMPORTANT NOTE: The following security relevant attributes are available as of SecurityServer 4.30 (CXI firmware module version 2.4.0.0 and later).	

Attribute	Description
CKA_CFG_ENFORCE_EXT_KEYS	If CKA_CFG_ENFORCE_EXT_KEYS (default: CK_FALSE) is set to CK_TRUE, an object (for example, a cryptographic key) is always created in the external keystore. A cryptographic key is created by the following actions: generate a key, generate a key pair, derive a key, restore a key, import a key and unwrap a key. CKA_CFG_ENFORCE_EXT_KEYS is irrelevant for key usage functions. For example, signing with an already existing internal key is supported even if CKA_CFG_ENFORCE_EXT_KEYS is set to CK_TRUE. If CKA_CFG_ENFORCE_EXT_KEYS is set to CK_TRUE, the KeysExternal parameter in the cs_pkcs11_R3.cfg file must be set to true and the KeyStorageType parameter and the KeyStorageConfig parameter in the same file must be set as well. See Chapter "Editing the cs_pkcs11_R3.cfg Configuration File" in [CS_PKCS11CAT (p. 99)] for details. If CKA_CFG_ENFORCE_EXT_KEYS is set to CK_TRUE and KeysExternal is set to false, no cryptographic key but an error message is generated.
CKA_CFG_ALLOW_WEAK_DES_KEYS	If CKA_CFG_ALLOW_WEAK_DES_KEYS (default: false) is set to true, importing weak DES keys is supported. A weak DES key is a 2DES key that has two equal parts or a 3DES key that has two parts that are pairwise equal.

Table 21: List of CryptoServer global configuration attributes

5.5 GetSlotConfig

See the table in chapter SetGlobalConfig (p. 70) for the list of available attributes and values.

This command displays the value of the slot configuration object attribute with the given name.

Syntax	`p11tool2 [Slot=<slot_id>] <login_command> GetSlotConfig=<attribute>`

Parameter	***Description***
`<slot_id>`	ID of the slot as number (to open a session). Default: 0
`<login_command>`	▪ Login as SO (via `LoginSO`) or ▪ Login as normal user (via `LoginUser`) ▪ Login as key manager or key user (via `Login`)
`<attribute>`	Name of the slot configuration attribute or '*' to get all slot configuration attribute values.

Example	`p11tool2 LoginUser=ask GetSlotConfig=CKA_CFG_WRAP_POLICY`

Output	`CKA_CFG_WRAP_POLICY = CK_FALSE`

5.6 SetSlotConfig

See the table in chapter SetGlobalConfig (p. 70) for the list of available attributes and values.

This command sets the value of the slot configuration object attribute with the given name to the given value.

The global configuration object attribute `CKA_CFG_ALLOW_SLOTS` must be set to `CK_TRUE`.

Only an SO is allowed to change the attributes of the slot configuration object. These attributes are the same as for the global configuration objec, except for the `CKA_CFG_ALLOW_SLOTS` attribute.

Syntax	p11tool2 [Slot=<slot_id>] LoginSO=<so_pin> SetSlotConfig=<attribute>,<value>

Parameter	***Description***
`<slot_id>`	ID of the slot as number (to open a session). Default: 0
`<so_pin>`	SO PIN in clear text or string 'ask' if hidden PIN entry is preferred.
`<attribute>`	Name of the slot configuration attribute
`<value>`	Value of the slot configuration attribute to be set

Example	`p11tool2 LoginSO=ask SetSlotConfig=CKA_CFG_WRAP_POLICY,CK_TRUE`

Output	None on success, or error message

5.7 SecureSlotPass

This command sets the optional, but recommended, passphrase to be used for the derivation of a slot-individual backup key as the `CKA_CFG_SLOT_BACKUP_HASH_PASS` slot configuration attribute.

The slot-individual backup key is not used by default, but only if the `CKA_CFG_SECURE_SLOT_BACKU`P configuration attribute is set to CK_TRUE. It is very important to first set the required passphrase (`CKA_CFG_SLOT_BACKUP_PASS_HASH`) and then enable the device to use slot-individual backup keys for protecting external key databases and their key backups created with P11CAT and p11tool2.

Changing the `CKA_CFG_SLOT_BACKUP_PASS_HASH` configuration attribute for a slot that is currently in use causes previously generated external key databases and their key backups to become inaccessible.

If you use SecurityServer/CryptoServer SDK 4.10 when creating the external key and their backups and you try to restore them with a changed individual passphrase, the error message “invalid mac of key blob” (error code: 0xB0680026) is created.

This applies as well if you have upgraded to SecurityServer/CryptoServer SDK 4.20 or later before trying to restore the external key or key backups.

However, if you upgrade to SecurityServer/CryptoServer SDK 4.20 before creating the external key or key backup and you try to restore them with a changed individual passphrase, the error message “wrong TBK passphrase for this key blob” (error code: 0xB0680081) is created.

The `CKA_CFG_SLOT_BACKUP_HASH_PASS` configuration attribute can be set up independently from the configuration of the `CKA_CFG_ALLOW_SLOTS` attribute.

Only an SO is allowed to change the attributes of the slot configuration object. The specified passphrase is only used if the `CKA_CFG_SECURE_SLOT_BACKUP` attribute is set to `CK_TRUE.`

Syntax	`p11tool2 [Slot=<slot_id>] LoginSO=<so_pin> SecureSlotPass=<passphrase>`

Parameter	*Description*
<slot_id>	ID of the slot as number (to open a session). Default:
<so_pin>	SO PIN in clear text or the string 'ask' if hidden PIN entry is preferred.
<passphrase>	A passphrase string to be used for the derivation of the slot-individual backup key that is used for protecting external keystores and key backups. For a hidden passphrase entry, enter the string 'ask' instead of the passphrase in clear text.

Example	`p11tool2 Slot=1 LoginSO=ask SecureSlotPass=ask`

Output	none on success, or error message

To remove the currently used passphrase, enter an empty passphrase string, e.g.:

```
p11tool2 Slot=1 LoginSO=ask SecureSlotPass=
```

5.8 Separated Key Manager and Key User Role

In PKCS#11 the USER has, by default, the permissions/tasks to manage keys (create, delete, import, export, etc.) and to use them in cryptographic operations.

These tasks can also be split into two groups and assigned to different PKCS#11 users:

- Key manager (KM) with permission mask 00000020 to manage cryptographic keys
- Key user (KU) with the permission mask 00000002 to use cryptographic keys.

Splitting the key manager and the key user into two separate roles can be achieved by setting the global or local configuration object `CKA_CFG_AUTH_KEYM_MASK,` see SetGlobalConfig (p. 70).

Initialize the slot as usual using p11tool2, see InitToken (p. 24),or P11CAT, see *Setting up a Slot (Init Token)* in the CryptoServer – PKCS#11 P11CAT - Manual.

However, the key manager and the key user have to be created manually. For example, perform these steps for slot 1 to do so:

1. Create a key manager (user KM_0001) with the permission mask 0000020 and the attribute CXI_GROUP=SLOT_0001. This command requires authentication by a user with the user administrator role (minimum permissions 20000000). This key manager must be created out of the scope of the CryptoServer PKCS#11 API and tools with the CryptoServer's administration tools csadm.

For example, using csadm:

```
csadm Dev=3001@127.0.0.1 LogonSign=ADMIN,:cs2:cjo:USB0
AddUser=KM_0001,00000020{CXI_GROUP=SLOT_0001},hmacpwd,123456
```

2. Perform the analog step to create the USR_0001 user of slot 1.

For example, using csadm:

csadm Dev=3001@127.0.0.1 LogonSign=ADMIN,:cs2:cjo:USB0
AddUser=USR_0001,00000002{CXI_GROUP=SLOT_0001},hmacpwd,123456

Do not initialize the slot PIN using p11tool2 or P11CAT. The "Init PIN" step would create a USR_000<x> user with the permission mask of a key manager and a key user (00000022) instead of the permission mask of a key user (00000002).

Now, the key manager can log in as user `KM_0001` with the `C_Login` function and user type `CKU_CS_GENERIC`, see Login (p. 21)) to perform key management functions. The key user can log in as usual with user type `CKU_USER`.

6 Backup/Restore Commands

These backup/restore commands are proprietary and not part of the PKCS#11 standard. They only work with the CryptoServer PKCS#11 Library R3 which has vendor defined extensions. For further information, see Chapter "Vendor Defined PKCS#11 Extensions" in the [CS_PKCS11DEV (p. 99)] provided on the SecurityServer product CD here `\Documentation\Crypto_APIs\PKCS11_R3.`

If you use a CryptoServer for a special certification, such as FIPS or Common Criteria, the needed authentication status might differ. See the documentations for the certified CryptoServer for more details.

6.1 GetBackupInfo

This command displays information about the given backup file.

Syntax	`p11tool2  GetBackupInfo=<filename>`

Parameter	***Description***
`<filename>`	Name of the backup file

Example	`p11tool2 GetBackupInfo=C:/backup/internal_keys.bak`

Output	`BACKUP_INFO:`
	
	`  File version     : 2`
	`  Creation date    : 20130327 144454`
	`  Slot ID          : 0x00000000`
	`  Slot Description : CryptoServer Device 'PCI:0' -`
	`SLOT_0000`
	
	`  Object count     : 16 internal key(s)`

6.2 BackupInternalKeys

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

Perform the `csadm MBKListKeys` command to determine which Master Backup Key (MBK) is currently in use in MBK slot 3 by the CryptoServer. This MBK is used by the `p11tool2 BackupInternalKeys` command to encrypt the backup file to be generated. If the MBK in MBK slot 3 is the autogenerated MBK named `AUTO-GEN`, the `p11tool2 BackupInternalKeys` command cannot be performed. Import a different MBK into MBK slot 3 using the `csadm MBKImportKey` command described in [CSADMIN] (p. 99).

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 Chapter "Master Backup Key Rollover" in [CSADMIN (p. 99)] for details.

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

Syntax	`p11tool2 [Slot=<slot_id>] [Force=<force>] <login_key_manager> BackupInternalKeys=<filename>`

Parameter	***Description***
`<slot_id>`	ID of the PKCS#11 slot as number Default: 0
<force>	Boolean flag (0/1 or n/y) to overwrite file if already exists. Default: 0 (Cancel command if file already exists)

Parameter	*Description*
<login_key_manager>	Login as key manager (via Login). Note: By default the key manager and the key user have the same permission mask. In that case the normal user can also be logged in (via LoginUser).
<filename>	Name of the key backup file to be created

Example	`p11tool2 Slot=1 Login=keyM,ask BackupInternalKeys=C:/backup/internal_keys_p11slot1.bak`

Output	`16 internal key(s) backed up`

6.3 BackupExternalKeys

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

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

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 Chapter "Master Backup Key Rollover" in [CSADMIN (p. 99)] for details.

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

Syntax	`p11tool2 [Slot=<slot_id>] [Force=<force>] <login_key_manager> BackupExternalKeys=<filename>`

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

Parameter	*Description*
<filename>	Name of the key backup file to be created

Example	`p11tool2 Slot=1 Login=keyM,ask BackupExternalKeys=C:/backup/external_keys_p11slot1.bak`

Output	`2 external key(s) backed up`

6.4 BackupConfig

This command backs up the PKCS#11 slot configuration object.

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

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 Chapter "Master Backup Key Rollover" in [CSADMIN (p. 99)] for details.

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

Syntax	`p11tool2 [Slot=<slot_id>] [Force=<force>] LoginSO=<so_pin> BackupConfig=<filename>`

Parameter	*Description*
`<slot_id>`	ID of the slot as number Default: 0
<force>	Boolean flag (0/1 or n/y) to overwrite file if already exists. Default: 0 (Cancel command if file already exists)
<so_pin>	SO PIN or string 'ask' if hidden PIN entry should be used.
<filename>	Name of the configuration backup file to be created

Example	`p11tool2 Slot=1 LoginSO=ask BackupConfig=C:/backup/config.bak`

Output	`slot configuration object backed up`

6.5 RestoreInternalKeys

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

Syntax	`p11tool2 [Slot=<slot_id>] <login_key_manager> RestoreInternalKeys=<filename>`

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

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

Output	`16 internal keys restored to internal key store`

6.6 RestoreExternalKeys

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

Syntax	`p11tool2 [Slot=<slot_id>] <login_key_manager> RestoreExternalKeys=<filename>`

Parameter	*Description*
<slot_id>	ID of the slot as number Default: 0

Parameter	*Description*
<login_key_manager>	Login as key manager (via Login). NOTE: By default the key manager and the key user have the same permission mask. In that case the normal user can also be logged in (via LoginUser).
<filename>	Name of a previously generated key backup file

Example	`p11tool2 Slot=1 Login=keyM,ask RestoreExternalKeys=C:/backup/external_keys.bak`

Output	`2 external keys restored to external key store`

6.7 RestoreConfig

This command restores the slot configuration object from the given configuration backup file.

Syntax	`p11tool2 [Slot=<slot_id>] LoginSO=<so_pin> RestoreConfig=<filename>`

Parameter	*Description*
`<slot_id>`	ID of the slot as number Default: 0
`<so_pin>`	SO PIN or string 'ask' if hidden PIN entry should be used.
`<filename>`	Name of a previously generated configuration backup file

Example	`p11tool2 Slot=1 LoginSO=ask RestoreConfig=C:/backup/config.bak`

Output	`slot configuration object restored`

6.8 DeleteSO

This command deletes the SO (security officer).

Syntax	`p11tool2 [Slot=<slot_id>] Login=<admin_name>,<admin_auth_token> DeleteSO`

Parameter	***Description***
`<slot_id>`	ID of the slot as number Default: 0
<admin_name>	Name of a CryptoServer administrator with permission mask 0x20000000

Parameter	*Description*
<admin_auth_token>	Authentication token of the CryptoServer administrator with <admin_name>: ▪ Case password-based authentication: Administrator password or string 'ask' if hidden password entry should be used. ▪ Case signature-based authentication: Key specifier where the private part of the administrator key should be loaded from: • Smartcard specifier, e.g., ':cs2:cjo:USB0' See Chapter "Authentication Mechanisms" in [CSADMIN (p. 99)] or [CSMSADM (p. 99)] for details about authentication mechanisms. • RSA and ECDSA signature authentication The PIN pad has to be connected to the computer where p11tool2 is running (USB port) or to another computer (see the Chapter "Using a Local PIN Pad for a Remote CryptoServer" in [CSADMIN (p. 99)]). Example: ':cs2:cjo:USB0@123.123.123.123/mypwd' • RSA smartcard authentication The Chapter "Using a Local PIN Pad for a Remote CryptoServer" in [CSADMIN (p. 99)] does not apply here because the PIN pad must be connected directly to the CryptoServer PCIe card. See Chapter "Authentication Mechanisms" in [CSADMIN (p. 99)] or [CSMSADM] for further details. The smartcard specifier can be omitted. The comma after the user name can be omitted as well (recommended). For an automatic login, replace the smartcard specifier by #<PIN>. • keyfile[#password], e.g., 'my.key#pwd' If the keyfile is encrypted, hidden password entry is possible by entering string 'ask' as password.

Example	`p11tool2 Slot=1 Login=ADMIN,"C:\Program Files\Utimaco\SecurityServer\Administration\ADMIN.key" DeleteSO`

Output	none on success, or error message

6.9 RecryptExternalKeys

This command is used when an MBK rollover is performed. See Chapter "Master Backup Key Rollover" in [CSADMIN (p. 99)] for details about an MBK rollover. The `p11tool2 RecryptExternalKey` s command creates a backup with the specified filename and recrypts all available external cryptographic keys within the slot with the current MBK.

The `p11tool2 RecryptExternalKeys` command only recrypts external cryptographic keys if the old MBK and the new MBK are AES MBKs.

The `p11tool2 RecryptExternalKeys` command must be performed for each PKCS#11 slot containing cryptographic keys.

The `p11tool2 RecryptExternalKeys` command is proprietary and only works with CryptoServer PKCS#11 library R3.

Syntax	`p11tool2 [Slot=<slot_id>] [Force=<force>] <login_key_manager> RecryptExternalKeys=<filename>`

Parameter	***Description***
<slot_id>	ID of the PKCS#11 slot as number Default: 0 The PKCS#11 slot number must not be confused with the MBK slot number.

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

Example	Back up all keys being available in PKCS#11 slot 1 and recrypt them with the current MBK. The USR_0000 user is also a key manager. `p11tool2 Slot=1 Login=USR_0000,123456 RecryptExternalKeys=p11.pks.bak`

Output	Example: 5 external key(s) backed up 5 external key(s) recrypted `or error message`

7 Contact Address for Support Queries

You can reach us from Monday to Friday, 09.00 a.m. to 05.00 p.m.

Utimaco IS GmbH
Germanusstr. 4
52080 Aachen
Germany

RMA Query

If you need to send the device back to Utimaco IS GmbH, please open a new RMA case (Return Merchandise Authorization). We request that you use the following web address. RMA cases cannot be opened by email or phone.

https://support.hsm.utimaco.com/support/rma/new

Other Support Queries

- Mail (preferred contact method)
  support@utimaco.com[1]
  Attach the diagnostic information to your email.
- Web portal
  https://support.hsm.utimaco.com/support/cases/new/
  The diagnostic information will be requested in our response if necessary.
- By phone
  AMERICAS +1-844-UTIMACO (+1 844-884-6226)
  EMEA +49 800-627-3081
  APAC +81 800-919-1301
  The diagnostic information will be requested in our response if necessary.

1 mailto:support@utimaco.com

8 References

Reference	Title/Company	Document No.
[CSADMIN]	CryptoServer – csadm - Manual/Utimaco IS GmbH.	2009-0003
[CSMSADM]	CryptoServer – Administration Manual / Utimaco IS GmbH.	M010-0001-en
[CS_PKCS11CAT]	CryptoServer – PKCS#11 P11CAT - Manual / Utimaco IS GmbH.	M013-0001-en
[CS_PKCS11DEV]	PKCS#11 R3 Developer Guide/Utimaco IS GmbH.	2012-0007
[CS_PKCS11HON]	Learning PKCS#11 in Half a Day – Using the Utimaco HSM Simulator/Utimaco IS GmbH.	2015-0008
[PKCS11ICMS]	"PKCS #11 Cryptographic Token Interface Current Mechanisms Specification Version 2.40," Committee Specification 01, September 16, 2014/ OASIS Standard. Available: http://docs.oasis-open.org/pkcs11/pkcs11-curr/v2.40/cs01/pkcs11-curr-v2.40-cs01.html	
[CSTrSh]	CryptoServer Troubleshooting/Utimaco IS GmbH.	M011-0008-en