THALES

SafeNet ProtectToolkit-C 5.9

MANAGEMENT LIBRARIES PROGRAMMING GUIDE

Document Information

Product Version	5.9
Document Part Number	007-013682-007
Release Date	08 January 2020

Revision History

Revision	Date	Reason
Rev. A	08 January 2020	Initial release

CONTENTS

PREFACE: About the SafeNet ProtectToolkit-C Management Libraries Programmer's Manual

The ProtectToolkit-C implementation of PKCS #11 may be extended using the KMLIB set of high level management libraries. KMLIB effectively wraps around PKCS #11 to form a new API that has all the PKCS #11 functionality but is simpler to use. This functionality is an optional extension to the standard ProtectToolkit-C functionality.

The guide contains the following chapters:

> "KMLIB Sample Applications" on page 10

> "KMLIB Callback Prototypes Reference" on page 14

> "KMLIB Function Reference" on page 27

This preface also includes the following information about this document:

> "Gemalto Rebranding" below

> "Audience" on the next page

> "Document Conventions" on the next page

> "Support Contacts" on page 9

For information regarding the document status and revision history, see "Document Information" on page 2.

Gemalto Rebranding

In early 2015, Gemalto completed its acquisition of SafeNet, Inc. As part of the process of rationalizing the product portfolios between the two organizations, the SafeNet name has been retained. As a result, the product names for SafeNet HSMs have changed as follows:

Old product name	New product name
ProtectServer External 2 (PSE2)	SafeNet ProtectServer Network HSM
ProtectServer Internal Express 2 (PSI-E2)	SafeNet ProtectServer PCIe HSM
ProtectServer HSM Access Provider	SafeNet ProtectServer HSM Access Provider
ProtectToolkit C (PTK-C)	SafeNet ProtectToolkit-C
ProtectToolkit J (PTK-J)	SafeNet ProtectToolkit-J

Old product name	New product name
ProtectToolkit M (PTK-M)	SafeNet ProtectToolkit-M
ProtectToolkit FM SDK	SafeNet ProtectToolkit FM SDK

> **NOTE** These branding changes apply to the documentation only. The SafeNet HSM software and utilities continue to use the old names.

Audience

This document is intended for personnel responsible for maintaining your organization's security infrastructure. This includes SafeNet ProtectToolkit users and security officers, key manager administrators, and network administrators.

All products manufactured and distributed by Thales are designed to be installed, operated, and maintained by personnel who have the knowledge, training, and qualifications required to safely perform the tasks assigned to them. The information, processes, and procedures contained in this document are intended for use by trained and qualified personnel only.

It is assumed that the users of this document are proficient with security concepts.

Document Conventions

This document uses standard conventions for describing the user interface and for alerting you to important information.

Notes

Notes are used to alert you to important or helpful information. They use the following format:

> **NOTE** Take note. Contains important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss. They use the following format:

> **CAUTION!** Exercise caution. Contains important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury. They use the following format:

****WARNING** Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury.**

Command Syntax and Typeface Conventions

Format	Convention
bold	The bold attribute is used to indicate the following: > Command-line commands and options (Type **dir /p**.) > Button names (Click **Save As**.) > Check box and radio button names (Select the **Print Duplex** check box.) > Dialog box titles (On the **Protect Document** dialog box, click **Yes**.) > Field names (**User Name:** Enter the name of the user.) > Menu names (On the **File** menu, click **Save**.) (Click **Menu > Go To > Folders**.) > User input (In the **Date** box, type **April 1**.)
italics	In type, the italic attribute is used for emphasis or to indicate a related document. (See the *Installation Guide* for more information.)
<variable>	In command descriptions, angle brackets represent variables. You must substitute a value for command line arguments that are enclosed in angle brackets.
[**optional**] [<optional>]	Represent optional **keywords** or <variables> in a command line description. Optionally enter the keyword or <variable> that is enclosed in square brackets, if it is necessary or desirable to complete the task.
{**a**|**b**|**c**} {<a>|<b>|<c>}	Represent required alternate **keywords** or <variables> in a command line description. You must choose one command line argument enclosed within the braces. Choices are separated by vertical (OR) bars.
[**a**|**b**|**c**] [<a>|<b>|<c>]	Represent optional alternate keywords or variables in a command line description. Choose one command line argument enclosed within the braces, if desired. Choices are separated by vertical (OR) bars.

Support Contacts

If you encounter a problem while installing, registering, or operating this product, please refer to the documentation before contacting support. If you cannot resolve the issue, contact your supplier or Thales Customer Support.

Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable database of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone

The support portal also lists telephone numbers for voice contact (Contact Us).

CHAPTER 1: KMLIB Sample Applications

Three sample applications are provided with the product, together with a makefile to demonstrate how to build the samples. The sample applications are:

- Generate a DES key and an RSA key pair ("Sample1.c" on the next page)
- Export a DES key file for key backup ("Sample2.c" on page 12)
- Import a DES key to file for key recovery ("Sample3.c" on page 13)

Unix and Win32 versions are provided. The makefile will compile the three samples into executable files. Use the following commands to do this:

- Win32: **nmake -f nt.mak**
- Unix: **gmake -f Makefile**

Sample1.c

This sample shows how to generate 2 types of keys using KMLIB. These are:

- A DES3 secret key
- An RSA key pair

The following assumptions are made:

- slot 1 already exists in the HSM
- the SO and user pin are set to 9999 for slot 1

Use either the **ctbrowse** utility or the following **ctkmu** utility command to check that the tokens have been created.

ctkmu l -s1

Documentation for the utilities can be found in the following manuals if required.

ctbrowse: *ProtectToolkit-C Administration Guide*

ctkmu: *ProtectToolkit-C Programming Guide*

Sample2.c

This sample shows how to back up tokens using KMLIB. It uses the tokens created by **Sample1.c** to create the backup.

The tokens in the sample are backed up to a file named **backup.bin**.

The following assumptions are made:

> Wrap key secret_key_example exists in Slot 1. This is created when **Sample1.c** is run

> The User pin is set to 9999 for Slot 1

Sample3.c

This sample shows how to recover tokens using KMLIB. It restores the tokens from the backup file created by **Sample2.c**.

The following assumptions are made:

> Unwrap key secret_key_example exists in Slot 1. This is created when **Sample1.c** is run.

> Backup file backup.bin exists. This is created when **Sample2.c** is run

> The User pin is set to 9999 for Slot 1

So that it can be confirmed that the tokens have been restored from the backup file, delete public_key_example prior to running **Sample 3.c**.

Use the following **ctkmu** utility command to delete the file:

ctkmu d -s1 -n public_key_example

To check that the tokens have been restored, use either the **ctbrowse** utility or the following **ctkmu** utility command:

ctkmu l -s1

Documentation for the utilities can be found in the following manuals if required.

ctbrowse: *ProtectToolkit-C Administration Guide*

ctkmu: *ProtectToolkit-C Programming Guide*

CHAPTER 2: KMLIB Callback Prototypes Reference

This chapter contains descriptions of the following callback prototypes:

Example of Callback Prototype Use

The following code demonstrates callback prototype use using the PromptString callback.

```
/** Function prototype */

static CK_RV PromptString(
   const char *pszMessage,
   char *pBuf, CK_ULONG *pBufLen
);

/** Function implementation */

static CK_RV PromptString(
   const char* pszMessage,
   char* pBuf,
   CK_ULONG* pBufLen
);

{
   CK_RV rv = CKR_OK;
   printf("%s", pszMessage);
   ReadLine(pBuf, *pBufLen);
   return rv;
}
```

```
int main(void){

   :
   :

   KM_Callbacks_t km_callbacks;
   memset(&km_callbacks, 0, sizeof(km_callbacks));
   km_callbacks.promptString       = PromptString;

   :


}
```

UICB_PromptConfirmation_t

This function will prompt the user for confirmation of an action, as specified in pszMessage.

If cancellable is TRUE, user is given the option to cancel, as well as YES/NO. The choice is returned in pResult.

Synopsis

#include < uicallbacks.h >

```
CK_RV(* UICB_PromptConfirmation_t)(
   const char *             pszMessage,
   CK_BBOOL                 bCancellable,
   UICB_ConfirmResult_t *   pResult
);
```

Parameter	Description
pszMessage	Message used to prompt the user for their choice
bCancellable	Specifies whether to give user the option to cancel
pResult	Choice, as selected by the user

UICB_PromptForSmartCard_t

Prompt and wait for a smart card to be entered in the specified slot.

Synopsis

#include < uicallbacks.h >

```
CK_RV(* UICB_PromptForSmartCard_t)(
   CK_SLOT_ID          cardSlotId,
   const char *        pszMessage,
   const char *        pszPrompt
);
```

Parameters	Description
cardSlotId	ID of the slot to wait for the smart card.
pszMessage	The message to display.
pszPrompt	The prompt to display.

UICB_PromptInt32_t

This function will prompt the user for an integer, as specified in pszMessage.

The number entered is returned in pInt.

Synopsis

#include < uicallbacks.h >

```
CK_RV(* UICB_PromptInt32_t)(
   const char *        pszMessage,
   CK_ULONG *          pUlong
);
```

Parameter	Description
pszMessage	Message used to prompt the user for the integer.
pUlong	Pointer to a CK_ULONG, which upon successful completion of this function, will contain the integer as entered by the user.

UICB_PromptKeyComponent_t

Prompt for a key component from the user and verify its KCV. It is up to the library to convert the hex string entered as the key component to binary.

Synopsis

#include < uicallbacks.h >

```
CK_RV(* UICB_PromptKeyComponent_t)(
   int                 compNum,
   int                 numComps,
   char *              pKeyComp,
   int *               pKeyCompLen
);
```

Parameter	Description
compNum	The number of the key component to get.
numComps	The number of key components to get.
pKeyComp	Location to store the verified key component.
pKeyCompLen	Location to store the length of the key component.

UICB_PromptPin_t

This function will prompt the user for the PIN of the specified user on the given slot.

The result is returned in the pPin parameter. The number of CK_CHARs copied into pPin is returned in pPinLen, which is also used as input to the function. If the number of CK_CHARs entered is larger than the number passed in pPinLen, the function will return CKR_BUFFER_TOO_SMALL and the required size will be returned in pPinLen.

Synopsis

#include < uicallbacks.h >

```
CK_RV(* UICB_PromptPin_t)(
   CK_SLOT_ID          slotId,
   CK_BBOOL            fConfirm,
   CK_USER_TYPE        userType,
   CK_CHAR *           pPin,
   CK_ULONG *          pPinLen
);
```

Parameter	Description
slotId	ID of the slot for which to prompt for the PIN.
fConfirm	Flag to indicate whether or not to confirm entered PIN.
userType	User whose PIN should be prompted for.
pPin	Pointer to buffer to accept PIN as entered by the user.
pPinLen	Pointer to a CK_ULONG specifying the length of the buffer. Upon successful completion of the function, the CK_ULONG will contain either the number of bytes copied into pPin, or the length required to hold the entered PIN.

UICB_PromptString_t

This function will prompt the user for a string, as specified in pszMessage.

The result is returned in the pszBuf parameter. The number of CK_CHARs copied into pszBuf is returned in pBufLen, which is also used as input to the function. If the number of CK_CHARs entered is larger than the number passed in in pBufLen, the function will return CKR_BUFFER_TOO_SMALL and the required size will be returned in pBufLen.

Synopsis

#include < uicallbacks.h >

```
CK_RV(* UICB_PromptString_t)(
   const char *        pszMessage,
   char *              pBuf,
   CK_ULONG *          pBufLen
);
```

Parameter	Description
pszMessage	Message used to prompt the user for the string.
pBuf	Pointer to buffer to accept string as entered by the user.
pBufLen	Pointer to the length of the buffer to accept the entered string. Upon successful completion of the function, the CK_ULONG will contain either the number of bytes copied into pPin, or the length required to hold the entered string.

UICB_PromptTokenPin_t

Prompt for a PIN.

Synopsis

#include < uicallbacks.h >

```
CK_RV(* UICB_PromptTokenPin_t)(
   const char *pszMessage,
   CK_CHAR *pPIN,
   int *pPinLen
);
```

Parameter	Description
pszMessage	The message to display when prompting for the PIN.
pPIN	Location to hold the PIN entered.
pPinLen	Location to hold the length of the entered PIN.

UICB_ShowExportHeader_t

Show the current smart card batch name and user number.

Synopsis

#include < uicallbacks.h >

```
void(* UICB_ShowExportHeader_t)(
   const char *        pszBatchName,
   int                 custodianNumber
);
```

Parameter	Description
pszBatchName	The name of the current batch.
custodianNumber	Number of the current custodian.

UICB_ShowImportHeader_t

Show the current smart card batch name and user number.

Synopsis

#include < uicallbacks.h >

```
void(* UICB_ShowImportHeader_t)(
   const char *        pszBatchName,
   const char *        pszUserName
);
```

Parameter	Description
pszBatchName	The name of the current batch.
pszUserName	The name of the user who owns the card.

UICB_ShowKeyComponent_t

Display a key component and KCV.

Synopsis

#include < uicallbacks.h >

```
CK_RV(* UICB_ShowKeyComponent_t)(
   int compNum, int    numComps,
   const char *        pKeyComp,
   const char *        pKCV
);
```

Parameter	Description
compNum	The current key component number.
numComps	The number of components to display.
pKeyComp	The key component to display.
pKCV	The KCV of the component to display.

UICB_ShowMsg_t

This function will display a message to the user as specified in pszMessage.

Synopsis

#include < uicallbacks.h >

```
void(* UICB_ShowMsg_t)(
   UICB_MsgType_t      msgType,
   const char *        pszMessage
);
```

Parameter	Description
pszMessage	Message to display.
msgType	Type of message to display.

UICB_ShowSCBatchInfo_t

Show smart card batch information.

Synopsis

#include < uicallbacks.h >

```
CK_RV(* UICB_ShowSCBatchInfo_t)(
   const char *         pszBatchName,
   const unsigned long  timeCreated,
   const unsigned long  cardNumber,
   const unsigned long  numCustodians,
   const unsigned long  custodianNumber,
   const char *         pszUserName,
   const double         percentageOfEks
);
```

Parameter	Description
pszBatchName	Name of the batch that this card belongs to.
timeCreated	Time this card was created.
cardNumber	Card number within the batch of this card.
numCustodians	Number of custodians within the batch.
custodianNumber	Number of this custodian.
pszUserName	Name of the user that 'owns' this card.
percentageOfEks	Percentage of the entire eks that has been stored on this card.

CHAPTER 3: KMLIB Function Reference

This chapter contains descriptions of the following functions:

KM_SetCallbacks

Set the callbacks for KMLIB to use.

Synopsis

#include <kmlib.h>

```
KM_SetCallbacks (
   KM_Callbacks_t *pCallbacks
);
```

Parameter	Description
pCallbacks	Pointer to the structure containing the callbacks to set. See "KMLIB Callback Prototypes Reference" on page 14 for callback prototypes.

Returns

```
CKR_ARGUMENTS_BAD
CKR_OK
```

KM_GetCallbacks

Get callbacks currently used by KMLIB.

Synopsis

#include <kmlib.h>

```
KM_GetCallbacks (
   KM_Callbacks_t *    pCallbacks
);
```

Parameter	Description
pCallbacks	Location to store the callback structure used by KMLIB. See "KMLIB Callback Prototypes Reference" on page 14 for callback prototypes.

Returns

```
CKR_ARGUMENTS_BAD
CKR_OK
```

KM_GenerateSecretKey

Generate a secret key.

> **NOTE** This function uses the following callbacks:
> > "UICB_PromptString_t" on page 20
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_GenerateSecretKey (
   CK_SESSION_HANDLE   hSession,
   CK_KEY_TYPE         keyType,
   CK_SIZE             keySizeInBits,
   CK_ATTRIBUTE *      pTpl,
   CK_COUNT            tplSize,
   CK_COUNT            numComps,
   CK_OBJECT_HANDLE *  phKey
);
```

Parameter	Description
hSession	Handle to an open session.
keyType	The type of PKCS #11 key to generate. Examples are: > CKK_AES > CKK_CAST128 > CKK_DES > CKK_DES2 > CKK_DES3 > CKK_IDEA > CKK_RC2 > CKK_RC4 > CKK_GENERIC_SECRET

Parameter	Description
keySizeInBits	Size, in bits, of the key to generate. This is not needed for fixed length key types. The size ranges for the supported PKCS #11 key types are: > CKK_AES - 128, 192 or 256 bits > CKK_CAST128 - 8, 64 or 128 bits > CKK_DES - 64 bits > CKK_DES2 - 128 bits > CKK_DES3 - 192 bits > CKK_IDEA - 128 bits > CKK_RC2 - 8 to 1024 bits in 8 bit increments > CKK_RC4 - 8 to 2048 bits in 8 bit increments > CKK_GENERIC_SECRET - 8 to "Effectively Infinite" bits
pTpl	The attribute template of the new key.
tplSize	Number of attributes in pTpl.
numComps	The number of components to generate if XORable key components are required. This parameter should be set to 0 or 1 if component generation is not required.
phKey	Location to store the handle of the generated key.

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_RANDOM_NO_RNG
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY
CKR_TEMPLATE_INCOMPLETE
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_WRITE_PROTECTED
CKR_USER_NOT_LOGGED_IN
```

KM_GenerateKeyPair

Generate a key pair.

> **NOTE** This function uses the following callback:
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_GenerateKeyPair (
   CK_SESSION_HANDLE   hSession,
   CK_KEY_TYPE         keyType,
   CK_SIZE             keySizeInBits,
   CK_ATTRIBUTE *      pPublicKeyTpl,
   CK_COUNT            publicKeyTplSize,
   CK_ATTRIBUTE *      pPrivateKeyTpl,
   CK_COUNT            privateKeyTplSize,
   CK_OBJECT_HANDLE *  phPublicKey,
   CK_OBJECT_HANDLE *  phPrivateKey
);
```

Parameter	Description
hSession	Handle to an open session.
keyType	The type of key pair to generate. Options are: > CKK_RSA > CKK_DSA > CKK_DH
keySizeInBits	Size, in bits, of the key pair to generate. The size ranges for the supported key types are: > CKK_RSA - 512 to 4096 bits in 256 bit increments > CKK_DSA - 512 to 1024 bits in 64 bit increments > CKK_DH - 512 to 2048 bits in 256 bit increments
pPublicKeyTpl	The attribute template the public key will have.
publicKeyTplSize	The number of attributes in pPublicKeyTpl.
pPrivateKeyTpl	The attribute template the public key will have.
privateKeyTplSize	The number of attributes in pPrivateKeyTpl.
phPublicKey	Location to store the handle of the new public key.
phPrivateKey	Location to store the handle of the new private key.

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY
CKR_TEMPLATE_INCOMPLETE
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_WRITE_PROTECTED
CKR_USER_NOT_LOGGED_IN
```

KM_ModifyBoolAttrs

Toggles the Boolean attributes of an object.

> **NOTE** This function uses the following callback:
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_ModifyBoolAttrs (
   CK_SESSION_HANDLE   hSession,
   CK_CHAR *           pUserPin,
   CK_SIZE             userPinLen,
   CK_CHAR *           pSoPin,
   CK_SIZE             soPinLen,
   CK_OBJECT_HANDLE    hObj,
   CK_ATTRIBUTE_TYPE * pAttrs,
   CK_COUNT            numAttrs
);
```

Parameter	Description
hSession	Handle to an open session.
pUserPin	Token User's PIN. If setting the CKA_EXPORT attribute, then the Token SO PIN is required. In this case, the session is logged out, then the SO is logged in to perform the operation, and eventually the User is logged back in.
userPinLen	Length of the users PIN.
pSoPin	Token SO PIN. If setting the CKA_EXPORT attribute, then the Token SO PIN is required. In this case, the session is logged out, then the SO is logged in to perform the operation, and eventually the User is logged back in. If the SO PIN is not provided, and is required, then it is prompted for.
soPinLen	Length of the SO PIN.
hObj	Handle to the object whose attributes are to be toggled.
pAttrs	Array of attribute types to modify. Each attribute specified in the array will be toggled.
numAttrs	The number of attributes to be toggled.

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_SENSITIVE
CKR_ATTRIBUTE_TYPE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
```

```
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_PIN_EXPIRED
CKR_PIN_INCORRECT
CKR_PIN_LOCKED
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY_EXISTS
CKR_TEMPLATE_INCONSISTENT
CKR_USER_ALREADY_LOGGED_IN
CKR_USER_ANOTHER_ALREADY_LOGGED_IN
CKR_USER_NOT_LOGGED_IN
CKR_USER_PIN_NOT_INITIALIZED
CKR_USER_TOO_MANY_TYPES
CKR_USER_TYPE_INVALID
```

KM_ImportFromSC

Import objects from one or more smart cards.

> **NOTE** This function uses the following callbacks:
> > "UICB_PromptForSmartCard_t" on page 16
> > "UICB_ShowImportHeader_t" on page 23
> > "UICB_PromptTokenPin_t" on page 21
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_ImportFromSC (
   CK_SESSION_HANDLE   hSession,
   CK_SLOT_ID          cardSlotId,
   CK_OBJECT_HANDLE    hUnwrapKey,
   CK_ULONG            importVersion
);
```

Parameters	Description
hSession	Handle to an open session.
cardSlotId	ID of the smart card slot to read smart cards from.
hUnwrapKey	Handle to the unwrapping key. Set to CK_INVALID_HANDLE for multiple custodian import.
importVersion	Version of import data to process. One of: > 200 - import Cprov2 backup data > 300 - import Cprov3/PTKC3 backup data

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_SENSITIVE
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CANT_LOCK
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DATA_LEN_RANGE
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_FUNCTION_NOT_SUPPORTED
CKR_GENERAL_ERROR
```

```
CKR_HOST_MEMORY
CKR_KEY_FUNCTION_NOT_PERMITTED
CKR_KEY_HANDLE_INVALID
CKR_KEY_SIZE_RANGE
CKR_KEY_TYPE_INCONSISTENT
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_OPERATION_NOT_INITIALIZED
CKR_PIN_EXPIRED
CKR_PIN_INCORRECT
CKR_PIN_LOCKED
CKR_SESSION_CLOSED
CKR_SESSION_COUNT
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_PARALLEL_NOT_SUPPORTED
CKR_SESSION_READ_ONLY
CKR_SESSION_READ_ONLY_EXISTS
CKR_SESSION_READ_WRITE_SO_EXISTS
CKR_SLOT_ID_INVALID
CKR_TEMPLATE_INCOMPLETE
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_NOT_PRESENT
CKR_TOKEN_NOT_RECOGNIZED
CKR_TOKEN_WRITE_PROTECTED
CKR_UNWRAPPING_KEY_HANDLE_INVALID
CKR_UNWRAPPING_KEY_INVALID
CKR_UNWRAPPING_KEY_SIZE_RANGE
CKR_UNWRAPPING_KEY_TYPE_INCONSISTENT
CKR_USER_ALREADY_LOGGED_IN
CKR_USER_ANOTHER_ALREADY_LOGGED_IN
CKR_USER_NOT_LOGGED_IN
CKR_USER_PIN_NOT_INITIALIZED
CKR_USER_TOO_MANY_TYPES
CKR_USER_TYPE_INVALID
CKR_WRAPPED_KEY_INVALID
CKR_WRAPPED_KEY_LEN_RANGE
```

KM_ImportFromFile

Import objects from a file.

> **NOTE** This function uses the following callback:
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_ImportFromFile (
   CK_SESSION_HANDLE   hSession,
   const char *        pszFileName,
   CK_OBJECT_HANDLE    hUnwrapKey,
   CK_ULONG            importVersion
);
```

Parameter	Description
hSession	Handle to an open session.
pszFileName	Fully qualified path to the file to read from.
hUnwrapKey	Handle to the unwrapping key.
importVersion	Version of import data to process. One of: > 200 - import Cprov2 backup data > 300 - import Cprov3/PTKC3 backup data

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_SENSITIVE
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DATA_LEN_RANGE
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_KEY_FUNCTION_NOT_PERMITTED
CKR_KEY_HANDLE_INVALID
CKR_KEY_SIZE_RANGE
CKR_KEY_TYPE_INCONSISTENT
CKR_MECHANISM_INVALID
```

```
CKR_MECHANISM_PARAM_INVALID
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_OPERATION_NOT_INITIALIZED
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY
CKR_TEMPLATE_INCOMPLETE
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_WRITE_PROTECTED
CKR_UNWRAPPING_KEY_HANDLE_INVALID
CKR_UNWRAPPING_KEY_SIZE_RANGE
CKR_UNWRAPPING_KEY_TYPE_INCONSISTENT
CKR_USER_NOT_LOGGED_IN
CKR_WRAPPED_KEY_INVALID
CKR_WRAPPED_KEY_LEN_RANGE
```

KM_ImportFromScreen

Import a key from console as either encrypted parts OR clear components.

> **NOTE** This function uses the following callbacks:
> > "UICB_PromptTokenPin_t" on page 21
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_ImportFromScreen (
   CK_SESSION_HANDLE   hSession,
   CK_CHAR *           pszLabel,
   CK_KEY_TYPE         keyType,
   CK_SIZE             keySizeInBits,
   CK_ATTRIBUTE *      pTpl,
   CK_COUNT            tplSize,
   CK_COUNT            numComps,
   CK_OBJECT_HANDLE    hUnwrapKey,
   CK_BBOOL            isEncMultiPart
);
```

Parameter	Description
hSession	Handle to an open session.
pszLabel	Label to give to the resulting key.
keyType	The type of the resulting key. Options are: > CKK_AES > CKK_CAST128 > CKK_DES > CKK_DES2 > CKK_DES3 > CKK_IDEA > CKK_RC2 > CKK_RC4 > CKK_GENERIC_SECRET

Parameter	Description
keySizeInBits	Size, in bits, of the resulting key. This is not needed for fixed length key types. The size ranges for the supported key types are: > CKK_AES - 128, 192 or 256 bits > CKK_CAST128 - 8, 64 or 128 bits > CKK_DES - 64 bits > CKK_DES2 - 128 bits > CKK_DES3 - 192 bits > CKK_IDEA - 128 bits > CKK_RC2 - 8 to 1024 bits in 8 bit increments > CKK_RC4 - 8 to 2048 bits in 8 bit increments > CKK_GENERIC_SECRET - 8 to "Effectively Infinite" bits
pTpl	The attribute template the imported key will have.
tplSize	The number of attributes in pTpl.
numComps	The number of XORable components that need to be entered to create the resulting key. This parameter is ignored if hUnwrap is not CK_INVALID_HANDLE.
hUnwrapKey	Handle to the unwrapping key to use to decrypt the entered encrypted parts. This parameter should be CK_INVALID_HANDLE if entering XORable components.
isEncMultiPart	Flag indicating if the key is to be imported by more than one encrypted part. The decrypted parts are concatenated to get the final key. This parameter only applies if hUnwrapkey is not CK_INVALID_HANDLE, and the key type is one of: > CKK_DES2 > CKK_DES3

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_SENSITIVE
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DATA_INVALID
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_KEY_HANDLE_INVALID
CKR_KEY_SIZE_RANGE
```

```
CKR_KEY_TYPE_INCONSISTENTCKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_RANDOM_NO_RNG
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY
CKR_SLOT_ID_INVALID
CKR_TEMPLATE_INCOMPLETE
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_NOT_PRESENT
CKR_TOKEN_NOT_RECOGNIZED
CKR_TOKEN_WRITE_PROTECTED
CKR_UNWRAPPING_KEY_HANDLE_INVALID
CKR_UNWRAPPING_KEY_SIZE_RANGE
CKR_UNWRAPPING_KEY_TYPE_INCONSISTENT
CKR_USER_NOT_LOGGED_IN
CKR_WRAPPED_KEY_INVALID
CKR_WRAPPED_KEY_LEN_RANGE
```

KM_ImportFromPinPad

Import a key from components entered on a PIN Pad device.

> **NOTE** This function uses the following callback:
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_ImportFromPinPad (
   CK_SESSION_HANDLE   hSession,
   CK_CHAR *           pszLabel,
   CK_KEY_TYPE         keyType,
   CK_SIZE             keySizeInBits,
   CK_ATTRIBUTE *      pTpl,
   CK_COUNT            tplSize,
   CK_COUNT            numComps,
   CK_OBJECT_HANDLE *  phKey
);
```

Parameter	Description
hSession	Handle to an open session.
pszLabel	Label to give resulting key.
keyType	The type of the resulting PKCS #11 key. For example: > CKK_AES > CKK_CAST128 > CKK_DES > CKK_DES2 > CKK_DES3 > CKK_IDEA > CKK_RC2 > CKK_RC4 > CKK_GENERIC_SECRET

Parameter	Description
keySizeInBits	Size, in bits, of the resulting key. This is not needed for fixed length key types. The size ranges for the supported PKCS #11 key types. For example: > CKK_AES - 128, 192 or 256 bits > CKK_CAST128 - 8, 64 or 128 bits > CKK_DES - 64 bits > CKK_DES2 - 128 bits > CKK_DES3 - 192 bits > CKK_IDEA - 128 bits > CKK_RC2 - 8 to 1024 bits in 8 bit increments > CKK_RC4 - 8 to 2048 bits in 8 bit increments > CKK_GENERIC_SECRET - 8 to "Effectively Infinite" bits
pTpl	The attribute template the resulting key will have.
tplSize	The number of attributes in template.
numComps	The number of XORable components that need to be entered to create the resulting key. This must be >= 2.
phKey	Location to store the handle of the resulting key.

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_KEY_HANDLE_INVALID
CKR_KEY_SIZE_RANGE
CKR_KEY_TYPE_INCONSISTENT
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY
CKR_TEMPLATE_INCOMPLETE
```

```
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_WRITE_PROTECTED
CKR_USER_NOT_LOGGED_IN
```

KM_ImportP12File

Import a certificate and private key from a PKCS #12 file.

> **NOTE** This function uses the following callbacks:
> > "UICB_PromptTokenPin_t" on page 21
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_ImportP12File (
   CK_SESSION_HANDLE   hSession,
   CK_CHAR *           pszFileName,
   CK_ATTRIBUTE *      pPrivateKeyTpl,
   CK_COUNT            privateKeyTplSize,
   CK_ATTRIBUTE *      pCertTpl,
   CK_COUNT            certTplSize,
   CK_OBJECT_HANDLE *  phPrivateKey,
   CK_OBJECT_HANDLE *  phCert
);
```

Parameter	Description
hSession	Handle to an open session.
pszFileName	Fully qualified name of the file to import from.
pPrivateKeyTpl	The attribute template the private key will have. Must have all boolean attributes that need to be TRUE, as well as the label. No other attributes are required.
privateKeyTplSize	The number of attributes in pPrivateKeyTpl.
pCertTpl	The attribute template the certificate will have. Must have all boolean attributes that need to be TRUE, as well as the label. No other attributes are required.
certTplSize	The number of attributes in pCertTpl.
phPrivateKey	Location to hold the handle of the resulting private key.
phCert	Location to store the handle of the resulting certificate.

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
```

```
CKR_DATA_LEN_RANGE
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_ENCRYPTED_DATA_INVALID
CKR_ENCRYPTED_DATA_LEN_RANGE
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_KEY_FUNCTION_NOT_PERMITTED
CKR_KEY_HANDLE_INVALID
CKR_KEY_SIZE_RANGE
CKR_KEY_TYPE_INCONSISTENT
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_OPERATION_NOT_INITIALIZED
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY
CKR_SIGNATURE_INVALID
CKR_SIGNATURE_LEN_RANGE
CKR_TEMPLATE_INCOMPLETE
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_WRITE_PROTECTED
CKR_USER_NOT_LOGGED_IN
```

KM_ExportToSCwMethod

Export one or more objects to one or more smart cards. It allows users to select the method used to generate component keys.

> **NOTE** This function uses the following callbacks:
> > "UICB_PromptConfirmation_t" on page 16
> > "UICB_PromptForSmartCard_t" on page 16
> > "UICB_PromptInt32_t" on page 17
> > "UICB_ShowExportHeader_t" on page 22
> > "UICB_PromptString_t" on page 20
> > "UICB_PromptTokenPin_t" on page 21
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_ExportToSCwMethod (
   CK_SESSION_HANDLE   hSession,
   CK_CHAR *           pUserPIN,
   CK_SIZE             userPinLen,
   CK_OBJECT_HANDLE *  phWrapeeObjs,
   CK_COUNT            numWrapeeObjs,
   CK_OBJECT_HANDLE    hWrapKey,
   CK_SLOT_ID          cardSlotId,
   uint32              deriveMech
);
```

Parameter	Description
hSession	Handle to an open session.
pUserPIN	The Token's User PIN.
userPinLen	Length of user PIN number
phWrapeeObjs	Array of handles to objects that are to be exported.
numWrapeeObjs	Number of objects in phWrapeeObjs.
hWrapKey	Label of the wrapping key. For multiple custodian export, this parameter is set to CK_INVALID_HANDLE.
cardSlotId	ID of the smart card slot to write to.

Parameter	Description
deriveMech	Mechanism used to derive component keys: > KM_XOR_MECHANISM > KM_NOFM_MECHANISM

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_SENSITIVE
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DATA_INVALID
CKR_DATA_LEN_RANGE
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_INFORMATION_SENSITIVE
CKR_KEY_FUNCTION_NOT_PERMITTED
CKR_KEY_HANDLE_INVALID
CKR_KEY_NOT_WRAPPABLE
CKR_KEY_SIZE_RANGE
CKR_KEY_TYPE_INCONSISTENT
CKR_KEY_UNEXTRACTABLE
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_OPERATION_NOT_INITIALIZED
CKR_PIN_EXPIRED
CKR_PIN_INCORRECT
CKR_PIN_INVALID
CKR_PIN_LEN_RANGE
CKR_PIN_LOCKED
CKR_SESSION_CLOSED
CKR_SESSION_COUNT
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_PARALLEL_NOT_SUPPORTED
CKR_SESSION_READ_ONLY
CKR_SESSION_READ_ONLY_EXISTS
CKR_SESSION_READ_WRITE_SO_EXISTS
CKR_SLOT_ID_INVALID
CKR_TEMPLATE_INCOMPLETE
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_NOT_PRESENT
CKR_TOKEN_NOT_RECOGNIZED
CKR_TOKEN_WRITE_PROTECTED
```

```
CKR_USER_ALREADY_LOGGED_IN
CKR_USER_ANOTHER_ALREADY_LOGGED_IN
CKR_USER_NOT_LOGGED_IN
CKR_USER_PIN_NOT_INITIALIZED
CKR_USER_TOO_MANY_TYPES
CKR_USER_TYPE_INVALID
CKR_WRAPPING_KEY_HANDLE_INVALID
CKR_WRAPPING_KEY_SIZE_RANGE
CKR_WRAPPING_KEY_TYPE_INCONSISTENT
```

KM_ExportToSC

Export one or more objects to one or more smart cards.

> **NOTE** This function is superseded by KM_ExportToSCwMethod (see "KM_ExportToSCwMethod" on page 48).

Synopsis

#include <kmlib.h>

```
KM_ExportToSC (
   CK_SESSION_HANDLE   hSession,
   CK_CHAR *           pUserPIN,
   CK_SIZE             userPinLen,
   CK_OBJECT_HANDLE *  phWrapeeObjs,
   CK_COUNT            numWrapeeObjs,
   CK_OBJECT_HANDLE    hWrapKey,
   CK_SLOT_ID          cardSlotId
);
```

Parameter	Description
hSession	Handle to an open session.
pUserPIN	The Token's User PIN.
userPinLen	Length of user PIN number.
phWrapeeObjs	Array of handles to objects that are to be exported.
numWrapeeObjs	Number of objects in phWrapeeObjs.
hWrapKey	Label of the wrapping key. For multiple custodian export, this parameter is set to CK_INVALID_HANDLE. > KM_XOR_MECHANISM > KM_NOFM_MECHANISM
cardSlotId	ID of the smart card slot to write to.

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_SENSITIVE
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DATA_INVALID
CKR_DATA_LEN_RANGE
```

```
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_INFORMATION_SENSITIVE
CKR_KEY_FUNCTION_NOT_PERMITTED
CKR_KEY_HANDLE_INVALID
CKR_KEY_NOT_WRAPPABLE
CKR_KEY_SIZE_RANGE
CKR_KEY_TYPE_INCONSISTENT
CKR_KEY_UNEXTRACTABLE
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_OPERATION_NOT_INITIALIZED
CKR_PIN_EXPIRED
CKR_PIN_INCORRECT
CKR_PIN_INVALID
CKR_PIN_LEN_RANGE
CKR_PIN_LOCKED
CKR_SESSION_CLOSED
CKR_SESSION_COUNT
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_PARALLEL_NOT_SUPPORTED
CKR_SESSION_READ_ONLY
CKR_SESSION_READ_ONLY_EXISTS
CKR_SESSION_READ_WRITE_SO_EXISTS
CKR_SLOT_ID_INVALID
CKR_TEMPLATE_INCOMPLETE
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_NOT_PRESENT
CKR_TOKEN_NOT_RECOGNIZED
CKR_TOKEN_WRITE_PROTECTED
CKR_USER_ALREADY_LOGGED_IN
CKR_USER_ANOTHER_ALREADY_LOGGED_IN
CKR_USER_NOT_LOGGED_IN
CKR_USER_PIN_NOT_INITIALIZED
CKR_USER_TOO_MANY_TYPES
CKR_USER_TYPE_INVALID
CKR_WRAPPING_KEY_HANDLE_INVALID
CKR_WRAPPING_KEY_SIZE_RANGE
CKR_WRAPPING_KEY_TYPE_INCONSISTENT
```

KM_ExportToFile

Export an encrypted object set to a file.

> **NOTE** This function uses the following callback:
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_ExportToFile (
   CK_SESSION_HANDLE  hSession,
   CK_OBJECT_HANDLE * phWrapeeObjs,
   CK_COUNT           numWrapeeObjs,
   CK_OBJECT_HANDLE   hWrapKey,
   const char *       pszFileName
);
```

Parameter	Description
hSession	Handle to an open session.
phWrapeeObjs	Array of handles to objects that are to be exported.
numWrapeeObjs	Number of objects in the phWrapeeObjs array.
hWrapKey	Handle to the wrapping key.
pszFileName	Fully qualified path to the file to export to.

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_SENSITIVE
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DATA_LEN_RANGE
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_INFORMATION_SENSITIVE
CKR_KEY_FUNCTION_NOT_PERMITTED
CKR_KEY_HANDLE_INVALID
CKR_KEY_NOT_WRAPPABLE
CKR_KEY_SIZE_RANGE
```

```
CKR_KEY_TYPE_INCONSISTENT
CKR_KEY_UNEXTRACTABLE
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_OPERATION_NOT_INITIALIZED
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY
CKR_TEMPLATE_INCOMPLETE
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_WRITE_PROTECTED
CKR_USER_NOT_LOGGED_IN
CKR_WRAPPING_KEY_HANDLE_INVALID
CKR_WRAPPING_KEY_SIZE_RANGE
CKR_WRAPPING_KEY_TYPE_INCONSISTENT
```

KM_ExportToScreen

Export a key to the console in encrypted parts. At this stage, only symmetric keys can be exported using this function.

> **NOTE** This function uses the following callback:
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
CK_RV KM_ExportToScreen (
   CK_SESSION_HANDLE   hSession,
   CK_OBJECT_HANDLE    hWrapeeKey,
   CK_OBJECT_HANDLE    hWrapKey,
   CK_BBOOL            isEncMuliPart
);
```

Parameter	Description
hSession	Handle to an open session.
hWrapeeKey	Handle to the key to export.
hWrapKey	Handle to the wrapping key.
isEncMuliPart	Flag indicating if the key is to exported in one or more encrypted parts. This parameter only applies if the wrappee key one of: > CKK_DES2 - exported as two parts > CKK_DES3 - exported as three parts

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_SENSITIVE
CKR_ATTRIBUTE_TYPE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_KEY_HANDLE_INVALID
CKR_KEY_NOT_WRAPPABLE
CKR_KEY_SIZE_RANGE
CKR_KEY_UNEXTRACTABLE
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
```

```
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_USER_NOT_LOGGED_IN
CKR_WRAPPING_KEY_HANDLE_INVALID
CKR_WRAPPING_KEY_SIZE_RANGE
CKR_WRAPPING_KEY_TYPE_INCONSISTENT
```

KM_DisplaySCStatus

Display information about the smart card entered in the specified slot.

> **NOTE** This function uses the following callbacks;
> > "UICB_PromptForSmartCard_t" on page 16
> > "UICB_ShowSCBatchInfo_t" on page 26
> > "UICB_ShowMsg_t" on page 25

Synopsis

#include <kmlib.h>

```
KM_DisplaySCStatus  (
   CK_SLOT_ID           cardSlot
);
```

Parameter	Description
cardSlot	Card slot ID.

Returns

```
CKR_ARGUMENTS_BAD
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_OK
CKR_SESSION_CLOSED
CKR_SESSION_COUNT
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_PARALLEL_NOT_SUPPORTED
CKR_SESSION_READ_WRITE_SO_EXISTS
CKR_SLOT_ID_INVALID
CKR_TOKEN_NOT_PRESENT
CKR_TOKEN_NOT_RECOGNIZED
CKR_TOKEN_WRITE_PROTECTED
```

KM_EnumerateAttributes

Enumerate all attributes for an object, returning a complete template containing all the objects attributes.

Synopsis

#include <kmlib.h>

```
CK_RV KM_EnumerateAttributes (
   CK_SESSION_HANDLE   hSession,
   CK_OBJECT_HANDLE    hObj,
   CK_ATTRIBUTE *      pTpl,
   CK_SIZE *           pTplSize
);
```

Parameter	Description
hSession	Handle to valid session.
hObj	Handle to the object whose attributes are to be enumerated.
pTpl	Points to an array of attributes, which are to be filled out by the function. If this parameter is NULL, no attributes are copied into the array.
pTplSize	The size of the pTpl array. Upon successful completion of the function, this parameter will contain the number of attributes contained in hObj.

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_SENSITIVE
CKR_ATTRIBUTE_TYPE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_OBJECT_HANDLE_INVALID
CKR_OK
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
```

KM_ExportToken

Export a token for a specific device.

Synopsis

#include <kmlib.h>

```
CK_RV KM_ExportToken (
   CK_SESSION_HANDLE   hSession,
   CK_CHAR             serialNumber[CK_SERIAL_NUMBER_SIZE],
   CK_BYTE *           tokenData,
   CK_ULONG *          pTokenDataSize
);
```

Parameter	Description
hSession	Handle to a USER logged in session on the token to be exported.
serialNumber	The serial number of the destination device for which the token is being exported.
tokenData	Location to store the exported token data. If NULL is specified no data will be exported, however pTokenDataSize will still return the size of exported token data.
pTokenDataSize	The size of the tokenData buffer. Upon completion of the function, this parameter will contain the size of the exported data. If pTokenDataSize is too small, no data will be placed in tokenData.

Returns

```
CKR_ARGUMENTS_BADCKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_KEY_HANDLE_INVALID
CKR_KEY_NOT_WRAPPABLE
CKR_KEY_SIZE_RANGE
CKR_KEY_UNEXTRACTABLE
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_USER_NOT_LOGGED_IN
CKR_WRAPPING_KEY_HANDLE_INVALID
CKR_WRAPPING_KEY_SIZE_RANGE
CKR_WRAPPING_KEY_TYPE_INCONSISTENT
```

KM_ImportToken

Import exported token data previously imported with KM_ExportToken() At this stage, only symmetric keys can be exported via this function.

Synopsis

#include <kmlib.h>

```
CK_RV KM_ImportToken (
   CK_SESSION_HANDLEhSession,
   const CK_BYTE *tokenData,
   CK_ULONGtokenDataLen
);
```

Parameter	Description
hSession	Handle to a USER logged in session on the token to be imported.
tokenData	The token data to import.
tokenDataLen	The size of the tokenData buffer.

Returns

```
CKR_ARGUMENTS_BAD
CKR_ATTRIBUTE_READ_ONLY
CKR_ATTRIBUTE_TYPE_INVALID
CKR_ATTRIBUTE_VALUE_INVALID
CKR_BUFFER_TOO_SMALL
CKR_CRYPTOKI_NOT_INITIALIZED
CKR_DEVICE_ERROR
CKR_DEVICE_MEMORY
CKR_DEVICE_REMOVED
CKR_FUNCTION_CANCELED
CKR_FUNCTION_FAILED
CKR_GENERAL_ERROR
CKR_HOST_MEMORY
CKR_MECHANISM_INVALID
CKR_MECHANISM_PARAM_INVALID
CKR_OK
CKR_OPERATION_ACTIVE
CKR_SESSION_CLOSED
CKR_SESSION_HANDLE_INVALID
CKR_SESSION_READ_ONLY
CKR_TEMPLATE_INCOMPLETE
CKR_TEMPLATE_INCONSISTENT
CKR_TOKEN_WRITE_PROTECTED
CKR_UNWRAPPING_KEY_HANDLE_INVALID
CKR_UNWRAPPING_KEY_SIZE_RANGE
CKR_UNWRAPPING_KEY_TYPE_INCONSISTENT
CKR_USER_NOT_LOGGED_IN
CKR_WRAPPED_KEY_INVALID
CKR_WRAPPED_KEY_LEN_RANGE
```