

SafeNet ProtectToolkit-C 5.7

MANAGEMENT LIBRARIES PROGRAMMER'S MANUAL



Document Information

Product Version	5.7
Document Part Number	007-013682-004
Release Date	10 September 2018

Revision History

Revision	Date	Reason
Rev. A	10 September 2018	Initial release

Trademarks, Copyrights, and Third-Party Software

Copyright 2009-2018 Gemalto. All rights reserved. Gemalto and the Gemalto logo are trademarks and service marks of Gemalto and/or its subsidiaries and are registered in certain countries. All other trademarks and service marks, whether registered or not in specific countries, are the property of their respective owners.

Disclaimer

All information herein is either public information or is the property of and owned solely by Gemalto and/or its subsidiaries who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Gemalto's information.

This document can be used for informational, non-commercial, internal, and personal use only provided that:

- > The copyright notice, the confidentiality and proprietary legend and this full warning notice appear in all copies.
- > This document shall not be posted on any publicly accessible network computer or broadcast in any media, and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided "AS IS" without any warranty of any kind. Unless otherwise expressly agreed in writing, Gemalto makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Gemalto reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Gemalto hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Gemalto be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Gemalto does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemalto be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemalto products. Gemalto disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service, or loss of privacy.

All intellectual property is protected by copyright. All trademarks and product names used or referred to are the copyright of their respective owners. No part of this document may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, chemical, photocopy, recording or otherwise without the prior written permission of Gemalto.

CONTENTS

Preface: About the SafeNet ProtectToolkit-C Management Libraries Programmer's Manual	6
Customer Release Notes	6
Gemalto Rebranding	6
Audience	7
Document Conventions	7
Support Contacts	9
Chapter 1: KMLIB Sample Applications	10
Sample1.c	11
Sample2.c	12
Sample3.c	13
Chapter 2: KMLIB Callback Prototypes Reference	14
Example of Callback Prototype Use	14
UICB_PromptConfirmation_t	16
UICB_PromptForSmartCard_t	16
UICB_PromptInt32_t	17
UICB_PromptKeyComponent_t	18
UICB_PromptPin_t	19
UICB_PromptString_t	20
UICB_PromptTokenPin_t	21
UICB_ShowExportHeader_t	22
UICB_ShowImportHeader_t	23
UICB_ShowKeyComponent_t	24
UICB_ShowMsg_t	25
UICB_ShowSCBatchInfo_t	26
Chapter 3: KMLIB Function Reference	27
KM_SetCallbacks	28
KM_GetCallbacks	29
KM_GenerateSecretKey	30
KM_GenerateKeyPair	32
KM_ModifyBoolAttrs	34
KM_ImportFromSC	36
KM_ImportFromFile	38
KM_ImportFromScreen	40
KM_ImportFromPinPad	43
KM_ImportP12File	46
KM_ExportToSCwMethod	48
KM_ExportToSC	51

KM_ExportToFile	53
KM_ExportToScreen	55
KM_DisplaySCStatus	57
KM_EnumerateAttributes	58
KM_ExportToken	59
KM_ImportToken	60

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](#)
- > ["Glossary of terms" on page 1](#)

This preface also includes the following information about this document:

- > ["Customer Release Notes" below](#)
- > ["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](#).

Customer Release Notes

The customer release notes (CRN) provide important information about this release that is not included in the customer documentation. It is strongly recommended that you read the CRN to fully understand the capabilities, limitations, and known issues for this release. You can view or download the latest version of the CRN for this release at the following location:

http://www.securedbysafenet.com/releasenotes/ptk/crn_ptk_5-4.pdf

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
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 Gemalto 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: <ul style="list-style-type: none"> > 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.)
<i>italics</i>	In type, the italic attribute is used for emphasis or to indicate a related document. (See the <i>Installation Guide</i> 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> <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> <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 [Gemalto Customer Support](#).

Gemalto 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 Gemalto 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.gemalto.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 Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Gemalto Customer Support by telephone at [+1 410-931-7520](tel:+14109317520). Additional local telephone support numbers are listed on the support portal.

Email Support

You can also contact technical support by email at technical.support@gemalto.com.

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 I -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:

- > ["UICB_PromptConfirmation_t" on page 16](#)
- > ["UICB_PromptForSmartCard_t" on page 16](#)
- > ["UICB_PromptInt32_t" on page 17](#)
- > ["UICB_PromptKeyComponent_t" on page 18](#)
- > ["UICB_PromptPin_t" on page 19](#)
- > ["UICB_PromptString_t" on page 20](#)
- > ["UICB_PromptTokenPin_t" on page 21](#)
- > ["UICB_ShowExportHeader_t" on page 22](#)
- > ["UICB_ShowImportHeader_t" on page 23](#)
- > ["UICB_ShowKeyComponent_t" on page 24](#)
- > ["UICB_ShowMsg_t" on page 25](#)
- > ["UICB_ShowSCBatchInfo_t" on page 26](#)

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

```
#include < uicallbacks.h >

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

Description

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.

Parameters

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

```
#include < uicallbacks.h >

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

Description

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

Parameters

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

```
#include < uicallbacks.h >
```

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

Description

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

The number entered is returned in pInt.

Parameters

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

```
#include < uicallbacks.h >
```

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

Description

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.

Parameters

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

```
#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
);
```

Description

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.

Parameters

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

```
#include < uicallbacks.h >
```

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

Description

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.

Parameters

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

```
#include < uicallbacks.h >
```

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

Description

Prompt for a PIN.

Parameters

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

```
#include < uicallbacks.h >

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

Description

Show the current smart card batch name and user number.

Parameters

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

UICB_ShowImportHeader_t

```
#include < uicallbacks.h >

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

Description

Show the current smart card batch name and user number.

Parameters

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

UICB_ShowKeyComponent_t

```
#include < uicallbacks.h >
```

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

Description

Display a key component and KCV.

Parameters

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

```
#include < uicallbacks.h >

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

Description

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

Parameters

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

UICB_ShowSCBatchInfo_t

```
#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
);
```

Description

Show smart card batch information.

Parameters

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" on the next page](#)
- > ["KM_GetCallbacks" on page 29](#)
- > ["KM_GenerateSecretKey" on page 30](#)
- > ["KM_GenerateKeyPair" on page 32](#)
- > ["KM_ModifyBoolAttrs" on page 34](#)
- > ["KM_ImportFromSC" on page 36](#)
- > ["KM_ImportFromFile" on page 38](#)
- > ["KM_ImportFromScreen" on page 40](#)
- > ["KM_ImportFromPinPad" on page 43](#)
- > ["KM_ImportP12File" on page 46](#)
- > ["KM_ExportToSCwMethod" on page 48](#)
- > ["KM_ExportToSC" on page 51](#)
- > ["KM_ExportToFile" on page 53](#)
- > ["KM_ExportToScreen" on page 55](#)
- > ["KM_DisplaySCStatus" on page 57](#)
- > ["KM_EnumerateAttributes" on page 58](#)
- > ["KM_ExportToken" on page 59](#)
- > ["KM_ImportToken" on page 60](#)

KM_SetCallbacks

```
#include <kmlib.h>

KM_SetCallbacks (
    KM_Callbacks_t *pCallbacks
);
```

Description

Set the callbacks for KMLIB to use.

Parameters

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

```
#include <kmlib.h>

KM_GetCallbacks (
    KM_Callbacks_t *    pCallbacks
);
```

Description

Get callbacks currently used by KMLIB.

Parameters

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

```
#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
);
```

Description

Generate a secret key.

Parameters

Parameter	Description
hSession	Handle to an open session.
keyType	The type of PKCS #11 key to generate. Examples are: <ul style="list-style-type: none"> > CKK_AES > CKK_CAST128 > CKK_DES > CKK_DES2 > CKK_DES3 > CKK_IDEA > CKK_RC2 > CKK_RC4 > CKK_GENERIC_SECRET
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: <ul style="list-style-type: none"> > 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.

Parameter	Description
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.



NOTE This function uses the following callbacks:

- > ["UICB_PromptString_t" on page 20](#)
- > ["UICB_ShowMsg_t" on page 25](#)

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

```
#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
);
```

Description

Generate a key pair.

Parameters

Parameter	Description
hSession	Handle to an open session.
keyType	The type of key pair to generate. Options are: <ul style="list-style-type: none"> > 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: <ul style="list-style-type: none"> > 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.



NOTE This function uses the following callback:

> ["UICB_ShowMsg_t" on page 25](#)

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

```
#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
);
```

Description

Toggles the Boolean attributes of an object.

Parameters

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.



NOTE This function uses the following callback:

> ["UICB_ShowMsg_t" on page 25](#)

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

```
#include <kmlib.h>
```

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

Description

Import objects from one or more smart cards.

Parameters

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: <ul style="list-style-type: none"> > 200 - import Cprov2 backup data > 300 - import Cprov3/PTKC3 backup data



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](#)

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_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

```
#include <kmlib.h>
```

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

Description

Import objects from a file.

Parameters

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: <ul style="list-style-type: none"> > 200 - import Cprov2 backup data > 300 - import Cprov3/PTKC3 backup data



NOTE This function uses the following callback:

> ["UICB_ShowMsg_t" on page 25](#)

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

```
#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
);
```

Description

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

Parameters

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: <ul style="list-style-type: none"> > 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: <ul style="list-style-type: none"> > 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: <ul style="list-style-type: none"> > CKK_DES2 > CKK_DES3



NOTE This function uses the following callbacks:

- > ["UICB_PromptTokenPin_t" on page 21](#)
- > ["UICB_ShowMsg_t" on page 25](#)

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_INCONSISTENT
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_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

```
#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
);
```

Description

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

Parameters

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: <ul style="list-style-type: none"> > 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: <ul style="list-style-type: none"> > 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.



NOTE This function uses the following callback:

> ["UICB_ShowMsg_t" on page 25](#)

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

```
#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
);
```

Description

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

Parameters

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.



NOTE This function uses the following callbacks:

- > ["UICB_PromptTokenPin_t" on page 21](#)
- > ["UICB_ShowMsg_t" on page 25](#)

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

```
#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
);
```

Description

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

Parameters

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.
deriveMech	Mechanism used to derive component keys: <ul style="list-style-type: none"> > KM_XOR_MECHANISM > KM_NOFM_MECHANISM



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](#)

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

```
#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
);
```

Description

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

Parameters

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.



NOTE This function is superseded by KM_ExportToSCwMethod (see "[KM_ExportToSCwMethod](#)" on page 48).

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

```
#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
);
```

Description

Export an encrypted object set to a file.

Parameters

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.



NOTE This function uses the following callback:

> ["UICB_ShowMsg_t" on page 25](#)

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

```
#include <kmlib.h>

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

Description

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

Parameters

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: <ul style="list-style-type: none"> > CKK_DES2 - exported as two parts > CKK_DES3 - exported as three parts



NOTE This function uses the following callback:

> ["UICB_ShowMsg_t" on page 25](#)

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

```
#include <kmlib.h>

KM_DisplaySCStatus (
    CK_SLOT_ID          cardSlot
);
```

Description

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

Parameters

Parameter	Description
cardSlot	Card slot ID.



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](#)

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

```
#include <kmlib.h>

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

Description

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

Parameters

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

```
#include <kmlib.h>

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

Description

Export a token for a specific device.

Parameters

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_BAD
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_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

```
#include <kmlib.h>

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

Description

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

Parameters

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
```