certgate

cgMiddleware
platform independant

User Guide

Version 1.0

05/31/2016

Autor: certgate

Document History

Version	Date	Autor	Notes
1.0	31.05.2016	certgate	-

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

Content

0 About this Guide 4
1 Introduction to PKCS#11 6
1.1 Basic principles of PKCS#11 7
1.2 Workflows / Using cryptoki 13
2 Supported features 14
2.1 Unsupported functions 14
2.2 Supported functions & requirement specification 16
2.3 Supported mechanisms 36
3 Object template 38
3.1 Attribute 38
3.2 Object templates 42
3.3 Function specific templates 45
4 Differences to standard cryptoki specification 2
4.1 KeyGeneration 2
4.2 Import 2
5 Examples 1
5.1 Import private key and search associated public key 1

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

List of Figures

Figure 1: abstraction layer of cryptoki 6
Figure 2: session states........ 11
Figure 3: class hirarchy 40

List of Tables

Table 1: unsupported functions 15
Table 2: supported mechanisms 37
Table 3: RSA key import 2
Table 4: supported EC OIDs........ 2
Table 5: EC key import........ 3
Table 6: generate a keypair........ 1

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

0 About this Guide

This guide contains all the necessary information for application developers in order to use certgate's middleware with dedicated smartcards. It gives a basic instruction to PKCS11 and lists all deviations from the cryptoki-standard which are necessary to support all certgate products with pre-installed smartcard applets (e.g. cgCard or cgToken). This guide is devided as follows:

Chapter 1.

First we introduce you with some basic knownledge of PKCS11 and its underlying class layout

Chapter 2.

Then all PKCS11-functions supported by our middleware are listed. Each function implements pre-defined requirements engineered in accordance with cryptoki-specification. These are also listed identified by the following requirement syntax:

@CTXXX where **X** are **numbers** between **0-9**.

e.g. **@CT001**: *pInitArgs shall have value NULL or shall point to a structure of type CK_C_INITIALIZE_ARGS.*

Furthermore we show which cryptographic mechanisms are supported by our library.

Chapter 3.

Then function specific templates are given. Most of the functions communicating with the smartcard handle those templates in order to e.g. generate a key-pair stored on the card. A template combines several attributes where not all functions support all combinations. This chapter shows which attributes are mandatory and how they are combined to build different use cases (e.g. object import or object generation).

Chapter 4.

Followed by showing the differences to standard cryptoki-specification. In order to support all the features offered by the applet some deviations are necessary. In this part of the documentation we give you information and reasons about those differences.

Chapter 5.

At the end of this guide basic examples shall strengthen the understading of PKCS11 / the use of the middleware.

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

Who is this guide for?

For app developers who want to use native PKCS11 code to maintain smartcard-objects (key-pairs, secret keys, certificates, storage objects) or use on-card-cryptography. PKCS11 can be used platform independent (all systems which support c / c++). Our library is shipped platform specific as e.g. **.dll** or **.lib** (with separate headers) and can be dynamically loaded into your program. Loading the library is outside of scope of this documentation.

What typographical conventions are used?

Warning

Provides mandatory information which should always kept in mind

Note
Provides additional information on a topic, and emphasize important facts and considerations.

Tip
Inform about best practices and other recommendations.

Note
You should have some basic understanding about Public-Key cryptography, digital certificates, digital signature and Public Key Infrastructure (PKI) in order for you to understand the discussed topics.

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

1 Introduction to PKCS#11

PKCS#11 is the standardization of a platform independant API for cryptographic tokens – in short "cryptoki" which stands for cryptographic token interface. Whenever we speak of cryptoki we use it as a synonym and mean the specification (see [CRTK04]) of the standard (other documents often mean the API).

On a physical abstraction level the standard is located almost on the top of the layers. Figure 1 shows how cryptoki is integrated in our environment. The standard itself does not define what kind of security token is used. Our implementation of cryptoki exclusively accesses smartcards as shown in Figure 1.

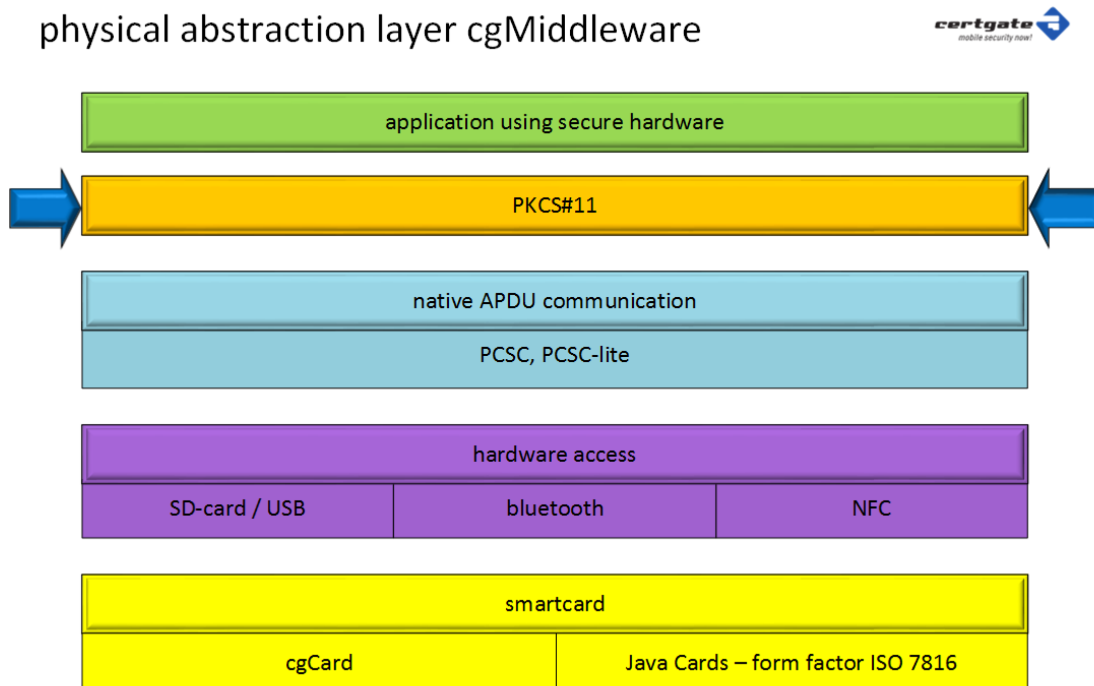

Figure 1: abstraction layer of cryptoki

Cryptoki has the goal to simplify the access of cryptographic hardware. Once developers have integrated the API and use it within their applications they can switch the middleware to a different PKCS#11 implementation and use the dedicated security token without chaning any line of code[1].

Before we list our supported parts of cryptoki (see chapter 2) we first want you to get familiar with some basic cryptoki principles. Developers who already got in contact with PKCS#11 can go on with reading the next chapter but are also invited to read the introduction parts.

[1] as far as both security tokens offer the same features and do not have distinct special behaviours

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

1.1 Basic principles of PKCS#11

We already learned that the cryptoki API can be universally used with different implementations for accessing different security tokens. In order to support this diversity some basics processes and features need to be specified. Within this part we specify those principles and show some of the most important workflows. Lets start with some basic notations.

1.1.1 Slot

Cryptoki supports multislot architecture where our implementation uses slots to identify smartcard readers. More generalized a slot can be seen as a hardware element having a security token inserted or not. Each slot has a unique ID which does not change as long as cryptoki is once initialized.

The slot ID will never change during the runtime of your application. Other applications could have different slot IDs - all IDs (not only slot IDs) are always context specific i.e. they are only valid for your application!

Slot-Info

Each slot has its own slot-info containing some details like manufacturer information, slotname and so on..

Slot-Events

As recently mentioned a slot will not always contain a token. As a matter of fact slots can be tagged as non removable (you can find this information in the slot info) i.e. that the security token – e.g. a smartcard cannot be removed because its solderd into the smartcard reader. All slots being tagged as removable should support slot-events. Slot-events give information about the reader state i.e. for example that a smartcard was inserted or removed.

1.1.2 Token

A Token can be seen as the piece of hardware[2] supporting the desired cryptographic features. It’s bound to the Slot ID which means that there is no own instance and it has no ID itself. Whenever you want to access the token you are just doing it virtually by accessing the Slot. Since a token can be removed / inserted there is the opportunity to get detailed information about it in order to e.g. avoid mix ups with other tokens.

Token-Info

Similar to the slot-info the token-info contains information about the manufacturer, model and so on. Furthermore this structure contains information about the tokens serial (for us – the serial number of the smartcard), available space, specific flags, etc. For us the flags are besides

[2] in fact a token is not always hardware – there could be software token implementations as well, e.g. a cryptograhic software keystore. Someone could also implement cryptoki in order to access those software elements.

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

the smart cards serial the most valuable information since they contain e.g. if the smartcard is locked due to too many invalid PIN retries.

1.1.3 Mechanism

As we mentioned earlier tokens can be – if tagged so – removed from the slot and even replaced by another one. Supported cryptographic features may vary between the tokens since applets can be easily replaced by offering new features. In a cryptoki specific context those supported features are called mechanisms. Since we do not want to implement another PKCS#11 for each released applet version cryptoki adds the support of dynamically reading the supported mechanisms of the currently inserted token. Mechanisms are splitted in eight different functionalities where we subclassed them to six different types

- Digests
  - Digest
- Ciphers
  - Encrypt / Decrypt,
  - Wrap / Unwrap
- Signatures
  - Sign / Verify
  - Sign Recover / Verify Recover
- KeyGenerators
  - Generate Key
- KeyPairGenerators
  - Generate KeyPair
- KeyDerivers
  - Derive

You can find our version-specific support of mechanisms in Table 2. A mechanism is identified by a pre-defined PKCS#11 ID. A token supporting a specific mechanism can further be asked for more details – called the mechanism-info.

Mechanism-Info

The mechanism-info gives information about the mechanism-class (in form of boolean values – isEncrypt, isDecrypt, …) and the token-specific supported key- or digests output-sizes.

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

1.1.4 Session

In order to access the tokens objects or to actively use its mechanisms a session needs to be opened. During a “normal” session – also known as public session – only temporary objects can be created. They are existent as long as the session is not closed[3] and are visible for all other open sessions created for this token. The keyword “normal” already implies that there are several session-states – so lets have a look at these and their limitations.

Session-state

The session-state limits the visibility of the token’s objects and manages its access rights e.g. whether an object can be persistently written to the smartcard. We distinguish the following states:

1. Public read mode

   In public read mode we can create / destroy temporary public non token objects – also called public session objects. Furthermore we can see all objects on the token which are tagged as public.

2. Public read / write mode

   In this mode we have the same features as in the previous mode. Furthermore we can persistently write public objects to the card such as a certificate, a public data object or a public key. We also can destroy those objects.

The applet installed on our smartcards supports the import of public keys BUT does not support internal calculations with them. This has the following reason:

Whenever a keypair (public / private key) is imported or generated the necessary information for doing cryptographical calculations is completely stored within the private key record. A public key record will be also written but this is just for completeness. On APDU level whether doing a public (encrypt / verify) or a private (decrypt / sign) calculation always needs the private key to be selected!

It is possible to store a single public key in a public key record BUT the applet will not be able to use it to encrypt or verify data. In order to support those calculations for single public keys or session objects (session objects are not stored on the token resulting in not being able to use the cards cryptographic processing unit) we offer this in software.

3. User read mode

   In this mode we have the same options as in the first mode. Furthermore we can read the private objects stored on the token and access them for cryptographic operations. Additionally we can create and destroy private session objects.

4. User read / write mode

   In this mode we have the same options as in the previous mode. Furthermore we have full access to create and destroy objects on the token. This mode is necessary to import or generate keypairs, secret keys or private data objects

[3] or the token is removed

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

5. SO mode

   In this mode we have the same options as in the first mode. Furthermore we can set (reset) the users PIN and install public root certificates (by tagging them with TRUSTED – this is not supported yet). We do NOT have any access to private objects but can create and destroy public token objects.

- In order to create a **user read session**, create a public read session and log in as user.
- In order to create a **user read / write session**, create a public read / write session and log in as user.
- In order to create a **SO session**, create a public read / write session and log in as SO.

See Figure 2 for visualization.

The mix-up of sessions having different session states is very limited:

1. When a user session exists no SO session can be created and vice versa
2. As soon as a session switches to user mode all open sessions switch to user mode
   ⇨ When a SO session exists no user session can be established due to 1.
3. A SO session can not be created out of a read session
4. As soon as a session switches to SO mode all open sessions switch to SO mode.
   ⇨ When a user session exists SO mode can NOT be established due to 1.
   ⇨ When a read session exists (and even no user is logged in) SO mode can NOT be established due to 3.

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

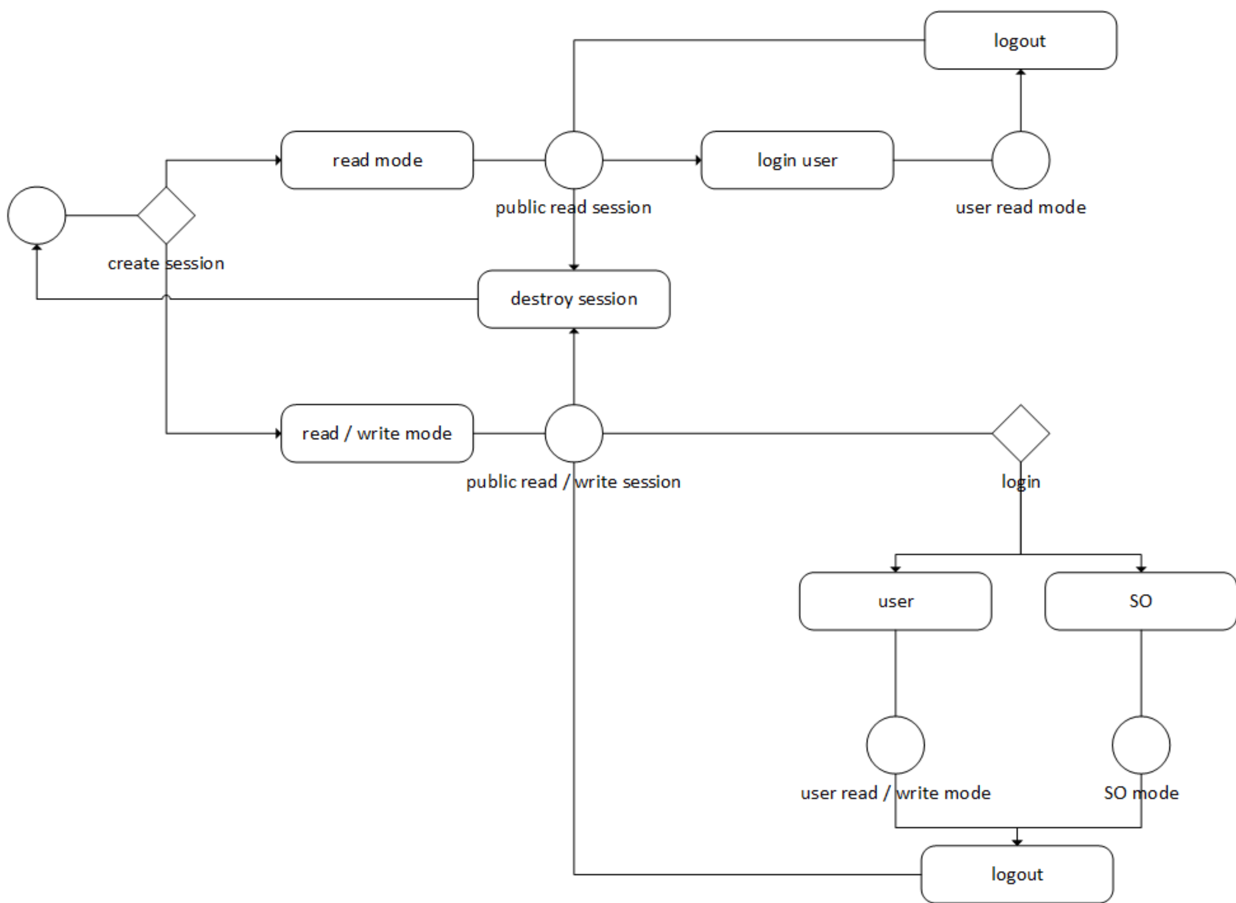

Figure 2: session states

Session objects

As we see objects stored on the token are accessible in different ways whereas session objects can be always created / destroyed[4].

A session object lives as long as its origin session stays alive and is accessible by all other sessions accessing that slot[5]. Other slot specific sessions are also allowed to destroy a session object even when they do not have created it!

During a public session only public session objects can be created / destroyed

During a user session public and private session objects can be created / destroyed

⇨ private session objects are never visible for public sessions since only user sessions can create private session objects and whenever a user is logged in all open sessions are converted to user sessions and are no more public.

[4] creating private session objects is still limited to user mode!

[5] in an application specific context - other applications may have their own session objects but do not have any access to foreign application specific created session objects.

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

1.1.5 Token Object

Contrary to session objects token objects are persistent and accessible to all applications[6]. Visibility and access rights are described in 1.1.4 (Session state). According to one object's type, objects are stored in different rom sections. We currently support to write the following types:

- Up to 32 private keys (RSA & EC)
- Up to 32 public keys (RSA & EC)
- Up to 255 certifcates[7] (X509)
- Up to 255 data objects[8]

In order to create a token object the "Token"-flag needs to be set to "true". For more information see 3.1.4 (Token). Otherwise a session object is created. Lets have a look at the supported types.

Key

A key can either be a secret-, a private- or a public key. The latter one needs not to be stored securely despite the others since their data is quite sensible. The smartcard's storage is exclusively separated for all these kind of keys where secret- and private keys are written to tamper resistant memory i.e. once created they will never leave the card. Public keys are written to simpler (and even cheaper) memory[9].

In order to distinguish those types the attributes "ClassType" and "KeyType" shall necessarily be written. For more information regarding these attributes see 3.1.4.

Certificate

Certificates are also stored in an own data section where we differ between trusted certificates and normal ones. Currently we only support the import of normal ones[10]. Trusted certificates have their own data section which is more secure than the normal rom storage.

Trusted certificates need a present "Trusted" attribute tagged with "true" where normal certificates do not have that attribute or are tagged with "false". This feature is not implemented yet.

Data

Data objects can either be stored as private or public. Private data objects can only be written or destroyed in user write sessions and just appear for user sessions. Public data objects are always visible and destroyable during write sessions.

[6] we previously learned that session objects are only valid in an application specifc context. When one application creates a token object all other applications using cryptoki may need to close all sessions and open one session again in order to detect a new token object.

[7] the applet supports up to 255 certificate entries. Due to variable certificate lengths the amount of entries varies and could be even less than this value.

[8] the same as mentioned in 7 applies for data objects

[9] in fact public keys are just written for completeness since they are also stored within the private key storage. See 1.1.4 Session state #2 for more information.

[10] whenever we speak of certificates we mean x509 certificates

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

1.2 Workflows / Using cryptoki

So far we've learned some PKCS#11 basics and are ready to exercise some workflows. As we already know we need a user read / write session in order to write private objects and at least a public read / write session to write public objects to the token. This small example tells us how to initialize the library and create a user read / write session. More information about e.g. how to gernerate or to import a keypair can be read in 3 and 5.

Before we can initialize the library we have to receive the PKCS#11 function addresses. This is done by calling **`C_GetFunctionList`** using an instance of **`CK_FUNCTION_LIST_PTR`** `as parameter.` After C_GetFunctionList returns successfully we can use the CK_FUNCTION_LIST_PTR for calling the PKCS#11 defined functions.

First we have to initialize the library by using the CK_FUNCTION_LIST_PTR to call C_Initialize. For this call we use NULL as parameter. The library then starts to communicate with the SCARD interface (also known as PCSC-lite) and retrieves all available smartcard information from the OS.

Now we have to call **`C_GetSlotList`** in order to get the available smartcard readers and their IDs. We can use CK_TRUE as first parameter in order to receive all slots having a smartcard inserted.

After receiving a valid slot having a token / smartcard inserted we can use its ID to create a session. For creating a session we call **`C_OpenSession`** having one previously received slotID and **`CKF_SERIAL_SESSION | CKF_RW_SESSION`** as two of the five parameters.

For our implementation it is mandatory to call C_OpenSession with **`CKF_SERIAL_SESSION`**

Now we are ready to change the session state from public read write to user read write by calling **`C_Login.`** Then we are ready for creating / destroying token objects or using the smartcards crypto functionality like encrypt[11].

We can revert all these steps by calling the following functions in the specified order:

- `C_Logout` to revert the user session back to a public session[12]
- `C_CloseSession` to destroy the session and all it's origin session objects[13]
- `C_Finalize` when cryptoki is not used anymore

[11] decrypt can even be called in public mode when using public key cryptography

[12] created private session objects will be destroyed, private token objects are no longer accessible. Other slot / token specific private sessions are also downgraded to public sessions.

[13] created token objects stay persistent

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

2 Supported features

In this chapter we focus on the supported features and functions implemented by our library. For a proper implementation of cryptoki all of the standard's specified functions have to be implemented where function stubs are allowed. These stubs shall return **CKR_FUNCTION_NOT_SUPPORTED.** We first start with listing these functions followed by our requirement specification for the ones which are implemented. At the end of this chapter we give an overview about the mechanisms currently implemented.

2.1 Unsupported functions

The following functions return with **CKR_FUNCTION_NOT_SUPPORTED** as long as no other error has higher priority (e.g. CK_CRYPTOKI_NOT_INITIALIZED).

Function Name	Reason (optional)	changed in version
C_InitToken		-
C_GetOperationState		-
C_SetOperationState		-
C_CopyObject	secret and private keys can not be copied, copying certificates and data objects is currently out of scope	-
C_EncryptUpdate	stream ciphers are currently not supported, use C_Encrypt instead (block cipher)	-
C_EncryptFinal	stream ciphers are currently not supported, use C_Encrypt instead (block cipher)	-
C_DecryptUpdate	stream ciphers are currently not supported, use C_Decrypt instead (block cipher)	-
C_DecryptFinal	stream ciphers are currently not supported, use C_Decrypt instead (block cipher)	-
C_DigestUpdate	use C_Digest instead	-
C_DigestKey	would only apply for session objects since a secret or private key can never leave the card again – out of scope	-
C_DigestFinal	use C_Digest instead	-
C_SignUpdate	MACs are currently not supported, use C_Sign instead (block cipher, sign / verify only cipher)	-

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

C_SignFinal	MACs are currently not supported, use C_Sign instead (block cipher, sign / verify only cipher)	-
C_SignRecoverInit		-
C_SignRevocer		-
C_VerifyUpdate	MACs are currently not supported, use C_Verify instead (block cipher, sign / verify only cipher)	-
C_VerifyFinal	MACs are currently not supported, use C_Verify instead (block cipher, sign / verify only cipher)	-
C_VerifyRecoverInit		-
C_VerifyRecover		-
C_DecryptEncryptUpdate		-
C_DecryptDigestUpdate		-
C_SignEncryptUpdate		-
C_DecryptVerifyUpdate		-
C_GenerateKey	secret keys are currently not supported	-
C_WrapKey	would only apply for session objects since a secret or private key can never leave the card again – out of scope	-
C_UnwrapKey		-
C_DeriveKey		-
C_SeedRandom	the hardware random number generator does not support seeding	-
C_GetFunctionStatus		-
C_CancelFunction		-
PKCS#11 function call-back		-

Table 1: unsupported functions

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

2.2 Supported functions & requirement specification

For quality purposes we analysed cryptoki and engineered requirements in accordance with the specification. Each requirement has an unique ID which was used for tagging our code wherever the requirement fulfilled cryptoki's needs. This allows us to track all implemented features and easyli maintain our code. The syntax of the IDs is as follows:

@CTXXX, where X is a number between 0-9

Green fields mean the requirement is completely implemented.

Red fields mean the requirement is not implemented.

Orange fields mean the requirement is partially implemented.

Not all functions have requirements specified.

2.2.1 C_Initialize

The function supports additional return values of:

- CKR_GENERAL_ERROR

req id	requirement description	changed in version
CT001	pInitArgs shall have value NULL or shall point to a structure of type CK_C_INITIALIZE_ARGS	-
CT002	pInitArgs shall be casted to a CK_C_INITIALIZE_ARGS_PTR when value is != NULL	-
CT003	pInitArgs->pReserved shall be NULL when pInitArgs has a value != NULL	-
CT004	when pInitArgs->pReserved and pInitArgs are both != NULL function shall return with CKR_ARGUMENTS_BAD	-
CT005	when CKF_LIBRARY_CANT_CREATE_OS_THREADS flag is set and application expects P11 lib beeing capable of multithreading function shall return CKR_NEED_TO_CREATE_THREADS	-
CT006	when CKF_OS_LOCKING_OK is not set and fields CreateMutex, DestroyMutex, LockMutex and UnlockMutex have value NULL P11 lib shall not use any multi-threading	-
CT007	when CKF_OS_LOCKING_OK is set and fields CreateMutex, DestroyMutex, LockMutex and UnlockMutex have value NULL P11 lib shall use OS primitives to ensure multithreaded safety	-
CT008	when CKF_OS_LOCKING_OK is not set and fields CreateMutex, DestroyMutex, LockMutex and UnlockMutex have value != NULL P11 lib shall use these function pointers to ensure mutithreaded safety	-
CT009	when CKF_OS_LOCKING_OK is set and fields CreateMutex, DestroyMutex, LockMutex and UnlockMutex have value != NULL P11 lib shall use either OS primitives or these function pointers to ensure multithreaded safety	-
CT010	when P11 lib is unable to ensure desired safe multithreaded access level function shall return CKR_CANT_LOCK	-

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

CT011	when CreateMutex, DestroyMutex, LockMutex and UnlockMutex partially have values != NULL function shall return CKR_ARGUMENTS_BAD	-
CT012	when pInitArgs is NULL function shall behave like CreateMutex, DestroyMutex, LockMutex, UnlockMutex, pReserved having value NULL and no flag being set	-
CT013	when function is called again and previously returned CKR_OK to the same application CKR_CRYPTOKI_ALREADY_INITIALIZED shall be returned	-
CT999	the function shall clear all currently available slots event states	-

2.2.2 C_Finalize

The function supports additional return values of:

- CKR_GENERAL_ERROR

req id	requirement description	changed in version
CT014	when pReserved parameter has value != NULL function shall return CKR_ARGUMENTS_BAD	-
CT015	when function is called without a preceding call of C_Initialize function shall return CKR_CRYPTOKI_NOT_INITIALIZED	-
CT016	when function is called all potentially waiting threads (which may called C_WaitForSlotEvent with enabled blocking) shall be unlocked	-

2.2.3 C_GetInfo

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_GENERAL_ERROR
- CKR_ARGUMENTS_BAD

2.2.4 C_GetFunctionList

The function supports additional return values of:

- CKR_GENERAL_ERROR
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT017	ppFunctionList shall receive the P11s references to the implemented api functions	-

2.2.5 C_GetSlotList

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

- CKR_GENERAL_ERROR
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT018	when pSlotList is NULL function shall set the number of current slots to out parameter pulCount and return with CKR_OK	-
CT019	when pSlotList is != NULL function shall return CKR_CKR_BUFFER_TO_SMALL when in parameter pulCount indicates that the out parameter pSlotList is not capable of storing all current slots and write the neccessary size to pulCount	-
CT020	when pSlotList is != NULL function shall insert all currently available slots to out parameter pSlotList, write the size to out parameter pulCount and return with CKR_OK	-
CT021	when tokenPresent is set to CK_TRUE function shall only return / count currently available slots that have a token present	-
CT022	unless function is called again with pSlotList = NULL all formerly reported slots shall be seen as valid slots (newly added slots are only accessible after calling this function with pSlotList = NULL again, slots which have been removed shall be still seen as valid as long as this function is called with pSlotList = NULL again)	-

2.2.6 C_GetSlotInfo

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_GENERAL_ERROR
- CKR_SLOT_ID_INVALID
- CKR_ARGUMENTS_BAD

2.2.7 C_GetTokenInfo

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_GENERAL_ERROR
- CKR_SLOT_ID_INVALID
- CKR_TOKEN_NOT_PRESENT
- CKR_ARGUMENTS_BAD

2.2.8 C_WaitForSlotEvent

The function supports additional return values of:

- CKR_GENERAL_ERROR

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

certgate

req id	requirement description	changed in version
CT023	when pReserved has a value != NULL function shall return with CKR_ARGUMENTS_BAD	-
CT024	when in parameter flag is set with CKF_DONT_BLOCK then the function shall set the ID of the reader where the most current event occured to the out parameter pSlot and return with CKR_OK	-
CT025	when in parameter flag is set with CKF_DONT_BLOCK and there are no pending events the function shall return with CKR_NO_EVENT	-
CT026	when in parameter flag has not set CKF_DONT_BLOCK the function shall wait until an event occurs	-
CT027	when the function is in waiting state and C_Finalize is called the function shall stop waiting for an event and return with CKR_CRYPTOKI_NOT_INITIALIZED	-
CT028	each currently accessable slot shall have an internal event flag which is set as soon as an event occurs	-
CT029	this function shall clear a slot’s event state whenever it reports the slot’s ID to the caller	-
CT030	this function shall react on token insertion event	-
CT031	this function shall react on token removal events	-

2.2.9 C_GetMechanismList

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_GENERAL_ERROR
- CKR_SLOT_ID_INVALID
- CKR_TOKEN_NOT_PRESENT
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT032	when pMechanismList is = NULL the function shall set the amount of the tokens supported mechanisms to pulCount out parameter and return with CKR_OK	-
CT033	when pMechanismList is != NULL function shall return CKR_CKR_BUFFER_TO_SMALL when in parameter pulCount indicates that the out parameter pMechanismList is not capable of storing all mechanisms supported by the token and write the neccessary size to pulCount	-
CT034	when pMechanismList is != NULL function shall insert all of the tokens supported mechanisms to out parameter pMechanismList and write its size to out parameter pulCount returning with CKR_OK	-

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

2.2.10 C_GetMechanismInfo

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_GENERAL_ERROR
- CKR_MECHANISM_INVALID
- CKR_SLOT_ID_INVALID
- CKR_TOKEN_NOT_PRESENT
- CKR_ARGUMENTS_BAD

2.2.11 C_InitPin

The function supports additional return values of:

- CKR_ CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_GENERAL_ERROR
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT035	the function shall initialize a normal users PIN	-
CT036	whenever the function is called not being in R/W SO Session state it shall return CKR_USER_NOT_LOGGED_IN	-
CT037	when the token has CKF_PROTECTED_AUTHENTICATION_PATH flag set parameter pPin shall have value = NULL	-

2.2.12 C_SetPin

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_GENERAL_ERROR
- CKR_PIN_INCORRECT
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT038	the function shall set the PIN of the currently logged in user	-
CT039	when no user is logged in at the moment and session is in R/W Public Session state the user PIN shall be set	-
CT040	whenever the function is called not being in a write state it shall return CKR_SESSION_READ_ONLY	-
CT041	when the token has CKF_PROTECTED_AUTHENTICATION_PATH flag set in parameters pOldPin and pNewPin shall have value = NULL	-

2.2.13 C_OpenSession

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_SLOT_ID_INVALID
- CKR_TOKEN_NOT_PRESENT
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT042	whenever the function is called without CKF_SERIAL_SESSION being set in flags input parameter the function shall return with CKR_SESSION_PARALLEL_NOT_SUPPORTED	-
CT043	whenever the function is called and the maximum amount of sessions is reached (token dependant) the function shall return with CKR_SESSION_COUNT	-
CT044	whenever a R/W SO Session is already open and a R Session is requested the function shall return with CKR_SESSION_READ_WRITE_SO_EXISTS	-

2.2.14 C_CloseSession

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_ DEVICE_REMOVED
- CKR_ FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

req id	requirement description	changed in version
CT045	all objects created during the session indicated by hSession shall be destroyed even when they are still be used by other sessions	-

2.2.15 C_CloseAllSessions

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_ FUNCTION_FAILED
- CKR_ GENERAL_ERROR
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT199	whenever the function is called all sessions opened for the slot indicated in slotID shall be closed	-
CT200	whenever the function is called all session objects created for that slot shall be destroyed	-
CT201	on successful call of this function the login state shall return to public i.e. potential new sessions are created in public state	-

2.2.16 C_GetSessionInfo

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_ FUNCTION_FAILED
- CKR_ GENERAL_ERROR
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_ARGUMENTS_BAD

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

2.2.17 C_Login

The function supports additional return values of:

- CKR_ARGUMENTS_BAD
- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_PIN_LOCKED
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_USER_ALREADY_LOGGED_IN
- CKR_USER_ANOTHER_ALREADY_LOGGED_IN
- CKR_USER_TYPE_INVALID

req id	requirement description	changed in version
CT046	on successful login all existing sessions shall be upgraded from public sessions to private sessions (user specific)	-
CT047	when trying to login with CKR_CONTEXT_SPECIFIC the function shall return CKR_OPERATION_NOT_INITIALIZED on improper use	-
CT048	when trying to login the SO and there is a read only session the function shall return CKR_SESSION_READ_ONLY_EXISTS	-
CT049	when CKF_PROTECTED_AUTHENTICATION_PATH flag is set the input parameter pPin shall be NULL	-
CT050	on successful login the function shall return with CKR_OK	-
CT051	the function shall return with CKR_PIN_INCORRECT when access can not be granted	-
CT052	logging in shall only succeed when no active operation is ongoing (no crypto operation, no object finding operations, ...)	-
CT053	the function shall only be called once unless a logout occurs or a key is post-accessed flagged with CKA_ALWAYS_AUTHENTICATE	-

2.2.18 C_Logout

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_SESSION_CLOSED

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

- CKR_SESSION_HANDLE_INVALID
- CKR_USER_NOT_LOGGED_IN

req id	requirement description	changed in version
CT054	on successful logout all existing sessions shall be downgraded from private sessions to public sessions (user specific)	-
CT055	on successful logout all private object handles shall be invalid even on re-login	-
CT056	on successful logout all privatly created non-token objects shall be destroyed	-
CT057	logging out shall only succeed when no active operation is ongoing (no crypto operation, no object finding operations, ...)	-

2.2.19 C_CreateObject

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_USER_NOT_LOGGED_IN

req id	requirement description	changed in version
CT058	whenever a template is unsupported the function shall return without creating an object	-
CT059	when creating key-objects the CKA_LOCAL attribute shall be set to CK_FALSE (create object is always an import -> not generated by token)	-
CT060	when creating private- or secret-key objects the CKA_ALWAYS_SENSITIVE attribute shall be set to CK_FALSE	-
CT061	when creating private- or secret-key objects the CKA_NEVER_EXTRACTABLE attribute shall be set to CK_FALSE	-
CT062	during a read only session only session objects shall be createable	-
CT063	during a public session only public objects shall be createable	-

2.2.20 C_DestroyObject

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

- CKR_GENERAL_ERROR
- CKR_OBJECT_HANDLE_INVALID
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_SESSION_READ_ONLY

req id	requirement description	changed in version
CT069	during a read only session only session objects shall be destroyed	-
CT070	during a public session only public objects shall be destroyed	-

2.2.21 C_GetObjectSize

The function supports additional return values of:

- CKR_ARGUMENTS_BAD
- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OBJECT_HANDLE_INVALID
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID

2.2.22 C_GetAttributeValue

The function supports additional return values of:

- CKR_ARGUMENTS_BAD
- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OBJECT_HANDLE_INVALID
- CKR_SESSION_HANDLE_INVALID

req id	requirement description	changed in version
CT071	in / out parameter pTemplate shall point to attributes described by type, value and length	-

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

CT072	when an objects is flagged as SENSITIVE or UNEXTRACTABLE the attributes out length shall be set to -1 and return value shall be set to CKR_ATTRIBUTE_SENSITIVE	-
CT073	when an object does not contain the specified attribute the attributes out length shall be set to -1 and return value shall be set to CKR_ATTRIBUTE_INVALID	-
CT074	when an objects attribute is extractable but the attributes in parameter value is NULL then the exact length shall be set to the attributes out length	-
CT075	when an objects attribute is extractable and the attributes out value field is large enough (indicated by length) the value shall be copied to out value and the exact length value shall be written to out length	-
CT076	when an objects attribute is extractable and the attributes out value field is not large enough the attributes out length shall be set to -1 and return value shall be set to CKR_BUFFER_TO_SMALL	-
CT077	whenever the return value is modified the function shall go on with the next attribute	-
CT078	whenever an objects attribute is flagged with CKF_ARRAY_ATTRIBUTE and consists of attributes (an attribute contains an array of attributes) the function shall treat each attribute like specified (CT072-CT077)	-

2.2.23 C_SetAttributeValue

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_ATTRIBUTE_READ_ONLY
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OBJECT_HANDLE_INVALID
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_SESSION_READ_ONLY
- CKR_USER_NOT_LOGGED_IN
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT079	during a read only session only session objects shall be modified	-
CT080	whenever the template includes attributes which are not compatible with the object or existing attributes the function shall return with CKR_TEMPLATE_INCONSISTENT	-

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

2.2.24 C_FindObjectsInit

The function supports additional return values of:

- CKR_ARGUMENTS_BAD
- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OPERATION_ACTIVE
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID

req id	requirement description	changed in version
CT081	when in parameter ulCount has value 0 all objects shall be "found"	-
CT082	objects shall be found in a session specific manner (public session -> only public objects)	-

2.2.25 C_FindObjects

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OPERATION_NOT_INITIALIZED
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT083	the function shall compare the search patterns bytewise	-
CT084	object finding shall be initialized with preceding call of C_FindObjectsInit	-
CT085	when there are no more objects to find pulObjectCount parameter shall receive value 0 - when objects where found pulObjectCount shall receive the amount of found objects	-

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

2.2.26 C_FindObjectsFinal

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OPERATION_NOT_INITIALIZED
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID

req id	requirement description	changed in version
CT086	function shall finalize objects search in order to start a potentially new one	-

2.2.27 C_EncryptInit

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_KEY_FUNCTION_NOT_PERMITTED
- CKR_KEY_HANDLE_INVALID
- CKR_KEY_TYPE_INCONSISTENT
- CKR_MECHANISM_INVALID
- CKR_OPERATION_ACTIVE
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_USER_NOT_LOGGED_IN

req id	requirement description	changed in version
CT087	in parameter hKey, being used as encryption key, shall have a valid (CK_TRUE) CKA_ENCRYPT attribute	-

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

2.2.28 C_Encrypt

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OPERATION_NOT_INITIALIZED
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT088	when out parameter pEncryptedData is NULL the function shall set parameter pulEncryptedDataLen to the number of bytes which would suffice to hold the output and return with CKR_OK	-
CT089	when out parameter pEncryptedData is != NULL and pulEncrypedDataLen indicates that the buffer is large enough to hold the output the function shall copy the output to pEncryptedData, set the exact size to pulEncryptedDataLen and return with CKR_OK	-
CT090	when out parameter pEncryptedData is != NULL and pulEncrypedDataLen indicates that the buffer is NOT large enough to hold the output the function shall set the exact size of the output to pulEncryptedDataLen and return with CKR_BUFFER_TO_SMALL	-
CT091	the function shall be precalled by C_EncryptInit in order to initialize an encryption process	-
CT092	the function shall always terminate an encryption process except it returns with CKR_BUFFER_TO_SMALL or was just used for a length call (CT088)	-
CT093	the function shall only support single-part-operations and cannot be called to finish multi-part-operations	-
CT094	when mechanism specific input length constraints are not satisfied the function shall return with CKR_DATA_LEN_RANGE	

2.2.29 C_DecryptInit

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_KEY_FUNCTION_NOT_PERMITTED

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

- CKR_KEY_HANDLE_INVALID
- CKR_KEY_TYPE_INCONSISTENT
- CKR_MECHANISM_INVALID
- CKR_OPERATION_ACTIVE
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_USER_NOT_LOGGED_IN
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT106	in parameter hKey, being used as decryption key, shall have a valid (CK_TRUE) CKA_DECRYPT attribute	-

2.2.30 C_Decrypt

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OPERATION_NOT_INITIALIZED
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_USER_NOT_LOGGED_IN
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT107	when out parameter pData is NULL the function shall set parameter pulDataLen to the number of bytes which would suffice to hold the output and return with CKR_OK	-
CT108	when out parameter pData is != NULL and pulDataLen indicates that the buffer is large enough to hold the output the function shall copy the output to pData set the exact size to pulDataLen and return with CKR_OK	-
CT109	when out parameter pData is != NULL and pulDataLen indicates that the buffer is NOT large enough to hold the output the function shall set the exact size of the output to pulDataLen and return with CKR_BUFFER_TO_SMALL	-
CT110	the function shall be precalled by C_DecryptInit in order to initialize an decryption process	-

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

CT111	the function shall always terminate an decryption process except it returns with CKR_BUFFER_TO_SMALL or was just used for a length call (CT107)	-
CT112	the function shall only support single-part-operations and cannot be called to finish multi-part-operations	-
CT113	when the ciphertext cannot be decrypted because it has inappropriate length the function shall either return CKR_ENCRYPTED_DATA_INVALID or CKR_ENCRYPTED_DATA_LEN_RANGE	-

2.2.31 C_DigestInit

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_MECHANISM_INVALID
- CKR_OPERATION_ACTIVE
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_ARGUMENTS_BA

2.2.32 C_Digest

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OPERATION_NOT_INITIALIZED
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT125	when out parameter pDigest is NULL the function shall set parameter pulDigestLen to the number of bytes which would suffice to hold the output and return with CKR_OK	-
CT126	when out parameter pDigest is != NULL and pulDigestLen indicates that the	-

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

	buffer is large enough to hold the output the function shall copy the output to pDigest set the exact size to pulDigestLen and return with CKR_OK	
CT127	when out parameter pDigest is != NULL and pulDigestLen indicates that the buffer is NOT large enough to hold the output the function shall set the exact size of the output to pulDigestLen and return with CKR_BUFFER_TO_SMALL	-
CT128	the function shall be precalled by C_DigestInit in order to initialize a hash process	-
CT129	the function shall always terminate a hash process except it returns with CKR_BUFFER_TO_SMALL or was just used for a length call (CT125)	-
CT130	the function shall only support single-part-operations and cannot be called to finish multi-part-operations	-

2.2.33 C_SignInit

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_KEY_FUNCTION_NOT_PERMITTED
- CKR_KEY_HANDLE_INVALID
- CKR_KEY_TYPE_INCONSISTENT
- CKR_MECHANISM_INVALID
- CKR_OPERATION_ACTIVE
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_USER_NOT_LOGGED_IN

req id	requirement description	changed in version
CT138	in parameter hKey, being used as signing key, shall have a valid (CK_TRUE) CKA_SIGN attribute	-

2.2.34 C_Sign

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

- CKR_OPERATION_NOT_INITIALIZED
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_USER_NOT_LOGGED_IN
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT139	when out parameter pSignature is NULL the function shall set parameter pulDataLen to the number of bytes which would suffice to hold the output and return with CKR_OK	-
CT140	when out parameter pSignature is != NULL and pulSignatureLen indicates that the buffer is large enough to hold the output the function shall copy the output to pSignature set the exact size to pulSignatureLen and return with CKR_OK	-
CT141	when out parameter pSignature is != NULL and pulSignatureLen indicates that the buffer is NOT large enough to hold the output the function shall set the exact size of the output to pulSignatureLen and return with CKR_BUFFER_TO_SMALL	-
CT142	the function shall be precalled by C_SignInit in order to initialize an decryption process	-
CT143	the function shall always terminate a signing process except it returns with CKR_BUFFER_TO_SMALL or was just used for a length call (CT139)	-
CT144	the function shall only support single-part-operations and cannot be called to finish multi-part-operations	-

2.2.35 C_VerifyInit

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_KEY_FUNCTION_NOT_PERMITTED
- CKR_KEY_HANDLE_INVALID
- CKR_KEY_TYPE_INCONSISTENT
- CKR_MECHANISM_INVALID
- CKR_OPERATION_ACTIVE
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_ARGUMENTS_BAD

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

certgate

req id	requirement description	changed in version
CT152	in parameter hKey, being used as verifying key, shall have a valid (CK_TRUE) CKA_VERIFY attribute	-

2.2.36 C_Verify

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OPERATION_NOT_INITIALIZED
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT153	the function shall be precalled by C_VerifyInit in order to initialize a verification process	-
CT154	the function shall always terminate a verification process	-
CT155	the function shall only support single-part-operations and cannot be called to finish multi-part-operations	-
CT156	whenever a signature can be seen as invalid purely on the basis of its length the function shall return CKR_SIGNATURE_LEN_RANGE	-
CT157	whenever a signature is invalid (verification fails) the funtion shall return CKR_SIGNATURE_INVALID	-
CT158	whenever a signature is valid the function shall return with CKR_OK	-

2.2.37 C_GenerateKeyPair

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_MECHANISM_INVALID
- CKR_OPERATION_ACTIVE
- CKR_SESSION_CLOSED

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

- CKR_SESSION_HANDLE_INVALID
- CKR_SESSION_READ_ONLY
- CKR_TEMPLATE_INCOMPLETE
- CKR_USER_NOT_LOGGED_IN
- CKR_ARGUMENTS_BAD

req id	requirement description	changed in version
CT173	the function shall be able to generate two new objects on a token	-
CT174	the function shall never generate a single public or private key	-
CT175	whenever the function succeeds it shall always generate a public AND a private key	-
CT176	the function shall read the type of the objects being created from in parameter pMechanism carried in the CKA_KEY_TYPE attribute	-
CT177	whenever one of the in parameter templates supplies a CKA_KEY_TYPE attribute other than specified in CKA_KEY_TYPE attribute of in parameter pMechanism the function shall return with CKR_TEMPLATE_INCONSISTENT	-
CT178	whenever one of the in parameter templates supplies a CKA_CLASS attribute other than specified in CKA_CLASS attribute of in parameter pMechanism the function shall return with CKR_TEMPLATE_INCONSISTENT	-
CT179	whenever one of the in parameter templates is not supported the function shall fail and do not create any object	-
CT180	objects created by this function shall always receive the CKA_LOCAL attribute with a value of CK_TRUE	-

2.2.38 C_GenerateRandom

The function supports additional return values of:

- CKR_CRYPTOKI_NOT_INITIALIZED
- CKR_DEVICE_REMOVED
- CKR_FUNCTION_FAILED
- CKR_GENERAL_ERROR
- CKR_OPERATION_ACTIVE
- CKR_SESSION_CLOSED
- CKR_SESSION_HANDLE_INVALID
- CKR_ARGUMENTS_BAD

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

2.3 Supported mechanisms

Currently we support the mechanisms listed in Table 2.

Some mechanisms are completely in hardware – some others use software for public key cryptography. Private key operations are always in hardware.

Hardware-only operations such as key generation are not supported for session objects!

Mechanism Type	Mechanism	Supported Sizes	Hardware / Software	since Version
Cipher, Signature	CKM_RSA_PKCS	KeySizes: 512 – 2048	Token objects: private key operations in hardware, public key operations in software Session objects: software	1.0
Signature	CKM_ECDSA_SHA1 supported curves: see KeyPairGen *CKM_EC_KEY_PAIR_GEN*	KeySizes: 160 – 320	Token objects: private key operations in hardware, public key operations in software Session objects: software	1.0
Signature	CKM_SHA1_RSA_PKCS	KeySizes: 512 – 2048	Token objects: private key operations in hardware, public key operations in software Session objects: Software	1.0
Signature	CKM_SHA256_RSA_PKCS	KeySizes: 512 – 2048	Token objects: private key operations in hardware, public key operations in software Session objects: Software	1.0
Signature	CKM_SHA512_RSA_PKCS	KeySizes: 768 – 2048	Token objects: private key operations in hardware, public key operations in software Session objects: software	1.0
KeyPairGen	CKM_RSA_PKCS_KEY_PAIR_GEN	512 – 2048	hardware	1.0
KeyPairGen	CKM_EC_KEY_PAIR_GEN supported curves:	160 – 320	hardware	1.0

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

	brainpoolP160r1 *brainpoolP192r1* *brainpoolP224r1* *brainpoolP256r1* *brainpoolP320r1* *ansi-x962 prime192v1* *ansip224r1* *ansi-x962 prime256v1*			
Digest	CKM_MD5	-	software	1.0
Digest	CKM_SHA_1	-	software	1.0

Table 2: supported mechanisms

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

3 Object template

This chapter gives an overview about `CK_ATTRIBUTES` and their usage. PKCS#11 objects consist of those attributes while some of the latter are quite important. For example the “token”-attribute decides whether an object is a token object or just a session object which only lives temporary during the active session. This chapter starts with giving detailed information of those attributes followed by an explaination how to use the attributes on different functions.

3.1 Attribute

As we already learned PKCS#11 objects consists of attributes where some of them are very important because they determine how the objects is stored, handled, … . Therefore we distinguish the following attribute classes:

- global mandatory attributes
- object specific attributes
- optional object specific attributes

The first category of attributes are necessary for all objects where the resting two depend on the purpose. Before we start with having a deeper look on attributes we will discuss how one of it is composed.

3.1.1 Structure of an attribue

An attribute consists of the following three parts

- The **type** of the attribute

  PKCS#11 specifies pre-defined constants identifying the attribute or more precisely the value, i. e. the type gives information about how the content of the value can be interpreted.

- The **value** of the attribute

  the value contains the intended data

- The **length** of the attribute

  the amount of bytes necessary to store the value

3.1.2 Construction of an object

Now as we now how attributes are structured we can exemplary construct a PKCS#11 object.

For example a RSA public key is described by the public exponent and its modulus. For constructing such an object in PKCS#11 we would need the gobal mandatory attributes and the key specific attributes like `CKA_PUBLIC_EXPONENT` and `CKA_MODULUS`. The public exponent attribute would may look like the following:

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

TYPE -> `CKA_PUBLIC_EXPONENT`

VALUE -> `010001`

LENGTH -> `3`

An optional attribute would be `CKA_MODULUS_BITS` since it gives extra information about the size of the modulus but isn't needed because this information could be derived from the modulus itsself. PKCS#11 defines all possible constructions of objects which really would go beyond the scope of this document for listing all of them here. For this reason we just concentrate on the base types and will discuss only the ones we really need in order to support our mechanisms. So lets start with the base types before we go on with defining the global mandatory attributes contained in every object.

3.1.3 PKCS#11 base objects

The PKCS#11 base objects follow the hirarchy depicted in Figure 3. There are three base objects we need for our ongoing investigations:

- Data objects
- Key objects
- Certificate objects

Each kind of object has its own set of possible attributes where base class attributes are additionally inherited. For example the Data base class consists of the following attributes:

- CKA_CLASS
- CKA_TOKEN
- CKA_PRIVATE
- CKA_LABEL
- CKA_MODIFIABLE
- CKA_APPLICATION
- CKA_OBJECT_ID
- CKA_VALUE

A data object is able to store values for all of these attributes but it does not neccessariliy have to set data for each of these. The next sub chapter gives us an overview which attributes are mandatory for all objects and guarantees distinguishablity. Afterwards we learn which attributes are mandatory for our three base objects. Having learned all necessary basics we are then able to talk about object templates for different key implementations such as RSA or EC.

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

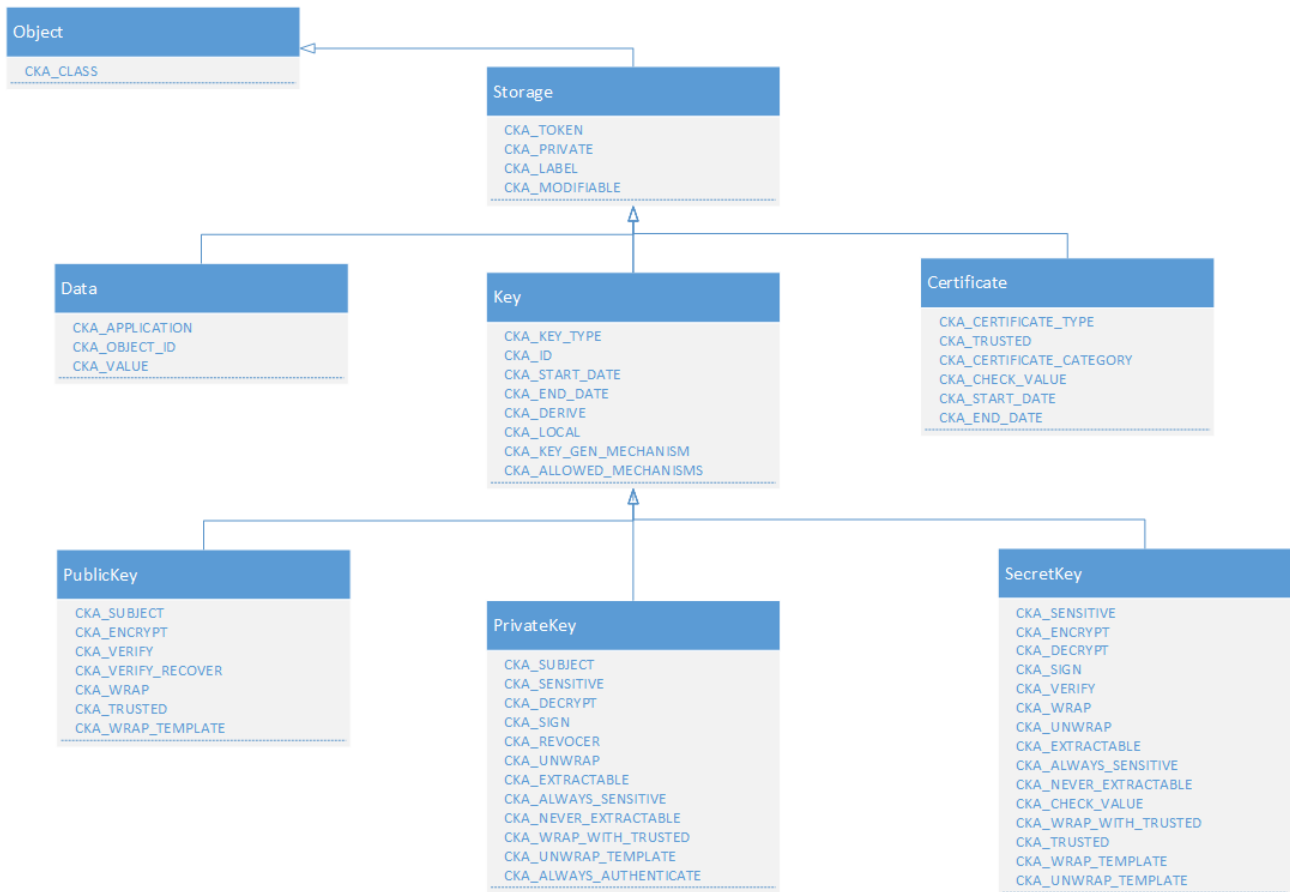

Figure 3: class hirarchy

3.1.4 Golbal mandatory attributes

An object, equal being a token-, a session-, or a data object (and so on..), always needs the global mandatory attributes being present. Not containing these attributes will result in errors – so without them no object can be written to or read from the token[14]. The mandatory attributes are:

- CKA_CLASS

  This attributes describes whether the object is a

  - Data object -> CKO_DATA
  - Certificate object -> CKO_CERTIFICATE
  - Public key object -> CKO_PUBLIC_KEY
  - Private key object -> CKO_PRIVATE_KEY
  - Secret key object -> CKO_SECRET_KEY

  There are also objects like CKO_HW_FEATURE, CKO_DOMAIN_PARAMETERS, CKO_MECHANISM, CKO_OTP_KEY and CKO_VENDOR_DEFINED but these are out of scope for this documentation.

- CKA_TOKEN

[14] this also applies for session objects

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

This attributes decides whether the object will be stored on the token or is a session object.

- CKA_PRIVATE

  This attribute decides whether a public session can read this object or the user must be logged in[15].

- CKA_MODIFIABLE

  This attribute decides whether the object can be changed after it was created.

Not all attributes can be changed. Currently we only support changing the attributes mentionend in 3.3.3.

The values set to the mandatory attributes influence the associated object for its complete lifetime.

3.1.5 Object specific attributes

We previously learned that the CKA_CLASS attribute is responsible for determining the objects kind. Having this information leads us to being more specific in enclosing the objects implementation type. Lets have a look back to Figure 3. Each of the data-, key- and certificate object contains one more attribute giving better information about what the object really is all about:

- Key contains CKA_KEY_TYPE
- Certificate contains CKA_CERTIFICATE_TYPE
- Data contains CKA_OBJECT_ID

These three attributes have one thing in common. They all give information about the implementation type of the object and which attributes it may include. Currently we support the following types:

- CKK_RSA and CKK_EC for CKA_KEY_TYPE

  In conjunction with the CKA_CLASS attribute this leads to the following implementation types:

  - RSA public key (CKA_CLASS -> CKO_PUBLIC_KEY)
  - RSA private key (CKA_CLASS -> CKO_PRIVATE_KEY)
  - EC public key (CKA_CLASS -> CKO_PUBLIC_KEY)
  - EC private key (CKA_CLASS -> CKO_PRIVATE_KEY)

- CKC_X_509 for CKA_CERTIFICATE_TYPE
- and any user specific OID for CKA_OBJECT_ID

[15] objects which should kept secret (like private- or secret keys) will internally overwrite this flag to private even when someone tries to create them as a public

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

Each of the implementation types is also defined in cryptoki. In 3.2 we list all additional attributes[16] supported by that implementation type and give information about the ones which are mandatory (object specific attributes) and those which are optional. Since cryptoki defines all possible implementation types we only handle the ones which are affected by our implementation.

3.2 Object templates

Regarding the implementation type each object has additionally attributes. Those attributes are defined here where the **fat** ones are mandatory while the others are optional. We first start with the base types before we handle the specific implementation types.

Each object inherits the base classe's attributes thus it also includes that attributes. Inherited attributes are no more listed and can be extracted from Figure 3.

3.2.1 Data base type

- **CKA_APPLICATION**
- CKA_OBJECT_ID
- CKA_VALUE

3.2.2 Key base type

- **CKA_KEY_TYPE**
- CKA_ID
- CKA_START_DATE (not yet supported for token objects)
- CKA_END_DATE (not yet supported for token objects)
- **CKA_DERIVE**
- CKA_LOCAL
- CKA_KEY_GEN_MECHANISM (not yet supported for token objects)
- CKA_ALLOWED_MECHANISMS (not yet supported for token objects)

3.2.3 Public key base type

- CKA_SUBJECT (not yet supported for token objects)
- **CKA_ENCRYPT**
- **CKA_VERIFY**
- CKA_VERIFY_RECOVER (not yet supported for token objects)

[16] the ones which are inherited are no more listed

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

- **CKA_WRAP**
- CKA_TRUSTED (not yet supported for token objects)
- CKA_WRAP_TEMPLATE (not yet supported for token objects)

3.2.4 Private key base type

- CKA_SUBJECT (not yet supported for token objects)
- **CKA_SENSITIVE**
- **CKA_DECRYPT**
- **CKA_SIGN**
- CKA_SIGN_RECOVER (not yet supported for token objects)
- **CKA_UNWRAP**
- **CKA_EXTRACTABLE** (not yet supported for token objects)
- **CKA_ALWAYS_SENSITIVE** (not yet supported for token objects)
- **CKA_NEVER_EXTRACTABLE** (not yet supported for token objects)
- CKA_WRAP_WITH_TRUSTED (not yet supported for token objects)
- CKA_UNWRAP_TEMPLATE (not yet supported for token objects)
- CKA_ALWAYS_AUTHENTICATE (not yet supported for token objects)

3.2.5 Secret key base type (not yet supported for token objects)

- **CKA_SENSITIVE**
- **CKA_ENCRYPT**
- **CKA_DECRYPT**
- **CKA_SIGN**
- **CKA_VERIFY**
- **CKA_WRAP**
- **CKA_UNWRAP**
- **CKA_EXTRACTABLE**
- **CKA_ALWAYS_SENSITIVE**
- **CKA_NEVER_EXTRACTABLE**
- CKA_CHECK_VALUE
- CKA_WRAP_WITH_TRUSTED
- CKA_TRUSTED
- CKA_UNWRAP_TEMPLATE

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

3.2.6 Certificate base type

- **CKA_CERTIFICATE_TYPE**
- CKA_TRUSTED (not yet supported for token objects)
- CKA_CERTIFICATE_CATEGORY (not yet supported for token objects)
- CKA_CHECK_VALUE (not yet supported for token objects)
- CKA_START_DATE (not yet supported for token objects)
- CKA_END_DATE (not yet supported for token objects)

3.2.7 RSA

Public key

- **CKA_MODULUS**
- CKA_MODULUS_BITS
- **CKA_PUBLIC_EXPONENT**
- CKA_VALUE (automatically calculated as ASN1 structure from modulus and exponent)

Private key

- CKA_MODULUS (automatically calculated from prime 1 and 2)
- **CKA_MODULUS_BITS (only mandatory for key generation)**
- **CKA_PUBLIC_EXPONENT**
- CKA_PRIVATE_EXPONENT (automatically calculated from public exponent)
- **CKA_PRIME_1**
- **CKA_PRIME_2**
- CKA_EXPONENT_1 (automatically calculated from prime 1)
- CKA_EXPONENT_2 (automatically calculated from prime 2)
- CKA_COEFFICIENT (automatically calculated from prime 1 and 2)

3.2.8 EC

Public key

- **CKA_EC_PARAMS**
- **CKA_EC_POINT (as BER-encoded uncompressed key)**
- CKA_VALUE (automatically calculated as ASN1 structure from ec-params and ec-point)

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

Private key

- **CKA_VALUE**
- **CKA_EC_PARAMS**
- CKA_EC_POINT (automatically calculated as BER-encoded uncompressed key)

3.2.9 X509 Certificate

- CKA_SUBJECT (not yet supported for token objects)
- CKA_ID
- CKA_ISSUER (not yet supported for token objects)
- CKA_SERIAL_NUMBER (not yet supported for token objects)
- **CKA_VALUE (ASN1 encoded certificate)**
- CKA_URL (not yet supported for token objects)
- CKA_HASH_OF_SUBJECT_PUBLIC_KEY (not yet supported for token objects)
- CKA_HASH_OF_ISSUER_PUBLIC_KEY (not yet supported for token objects)
- CKA_JAVA_MIDP_SECURITY_DOMAIN (not yet supported for token objects)

3.3 Function specific templates

In this section we handle object templates used for different use cases - these are:

- Create / import an object
- Generate a keypair
- Change attributes

We will learn how the attributes shall be used in order to support our cryptoki implementation since in most of the writing-cases (create, change, generate) there are always some token (for us applet-) dependant attributes which are mandatory even when the specification says they are not or attributes which can’t be modified or are unsupported.

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

3.3.1 Create / import an object

RSA

Table 3 describes the mandatory attributes for importing a RSA private key (token object) and a RSA public key (session object).

Since the applet generates the public key itself (for token objects) this object should not be created. Just import the private key and use the object-search (use the private keys CKA_ID) afterwards to find the corresponding public key. An example is given in 5.

RSA private key (token object)

During the import a label can be also given. If not CKA_LABEL receives the same value as CKA_ID.

The values for CKA_MODULUS, CKA_PRIVATE_EXPONENT, CKA_EXPONENT_1, CKA_EXPONENT_2 and CKA_COEFFICIENT can be also given. If not[17] they are calculated in software from CKA_PRIME_1, CKA_PRIME_2 and CKA_PUBLIC_EXPONENT.

Do not use CKA_MODULUS_BITS as attribute when creating / importing a RSA keypair. This attribute is reserved and is used for detecting keypair generation! The attribute will be automatically set during import.

We do not support the creation / import of a private key session object since we only allow public key operations in software.

[17] when one of the previous attributes is missing all these attributes are calculated

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

Since we only support public key operations in software a RSA private key can not be created as session object. Private key operations shall always be done in hardware – therefore import the private key as token object. RSA public keys can be created as session objects in order to encrypt[18] data for a receiver or to verify[18] a signature. An example is given in 5.

RSA public key (session object)

During the import a label can be also given. If not CKA_LABEL receives the same value as CKA_ID.

During the import CKA_MODULUS_BITS can be also given. If not it is calculated from CKA_MODULUS.

Key Type	CKA_CLASS	CKA_TOKEN	CKA_PRIVATE	CKA_MODIFIABLE	CKA_KEY_TYPE	CKA_ID	CKA_PRIME_1	CKA_PRIME_2	CKA_PUBLIC_EXPONENT
Private key (token object)	CKO_PRIVATE_KEY	CK_TRUE	CK_TRUE	CK_TRUE or CK_FALSE	CKK_RSA	All UTF8 symbols	CK_BYTE[]	CK_BYTE[]	CK_BYTE[] (recommended is 0x010001)
Key Type	**CKA_CLASS**	**CKA_TOKEN**	**CKA_PRIVATE**	**CKA_MODIFIABLE**	**CKA_KEY_TYPE**	**CKA_ID**	**CKA_MODULUS**	**CKA_PUBLIC_EXPONENT**	**-**
Public key (session object)	CKO_PUBLIC_KEY	CK_FALSE	CK_FALSE	CK_TRUE or CK_FALSE	CKK_RSA	All UTF8 symbols	CK_BYTE[]	CK_BYTE[]	-

Table 3: RSA key import

[18] according to the relevant access flags

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

EC

Table 5 describes the mandatory attributes for importing a EC private key (token object) and a EC public key (session object).

Table 4 lists the OIDs of the supported elliptic curves – use the desired OID as CK_EC_PARAMS.

Since the applet generates the public key itself (for token objects) this object should not be created. Just import the private key and use the object-search (use the private keys CKA_ID) afterwards to find the corresponding public key. An example is given in 5.

During the import a label can be also given. If not CKA_LABEL receives the same value as CKA_ID.

The value CKA_EC_POINT[19] can be also given. If not it is calculated in software from CKA_VALUE.

We do not support the import of a private key session object since we only allow public key operations in software.

[19] In uncompressed format => starting with 0x04 indicating this is the uncompressed format followed by concatenating the x and then the y coordinate of the point

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

Use one of the following OID values as CK_BYTE[] as value for CKA_EC_PARAMS attribute.

Defined elliptic curve	**OID**
brainpoolP160r1	0x06, 0x09, 0x2B, 0x24, 0x03, 0x03, 0x02, 0x08, 0x01, 0x01, 0x01
brainpoolP192r1	0x06, 0x09, 0x2B, 0x24, 0x03, 0x03, 0x02, 0x08, 0x01, 0x01, 0x03
brainpoolP224r1	0x06, 0x09, 0x2B, 0x24, 0x03, 0x03, 0x02, 0x08, 0x01, 0x01, 0x05
brainpoolP256r1	0x06, 0x09, 0x2B, 0x24, 0x03, 0x03, 0x02, 0x08, 0x01, 0x01, 0x07
brainpoolP320r1	0x06, 0x09, 0x2B, 0x24, 0x03, 0x03, 0x02, 0x08, 0x01, 0x01, 0x09
ansi-x962 prime192v1	0x06, 0x08, 0x2A, 0x86, 0x48, 0xCE, 0x3D, 0x03, 0x01, 0x01
ansip224r1	0x06, 0x05, 0x2B, 0x81, 0x04, 0x00, 0x21
ansi-x962 prime256v1	0x06, 0x08, 0x2A, 0x86, 0x48, 0xCE, 0x3D, 0x03, 0x01, 0x07

Table 4: supported EC OIDs

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

Since we only support public key operations in software an EC private key can not be created as session object. Private key operations shall always be done in hardware – therefore import the private key as token object. EC public keys can be created as session objects in order to verify[18] a signature. An example is given in 5.

EC public key (session object)

During the import a label can be also given. If not CKA_LABEL receives the same value as CKA_ID.

Key Type	CKA_CLASS	CKA_TOKEN	CKA_PRIVATE	CKA_MODIFIABLE	CKA_KEY_TYPE	CKA_ID	CKA_EC_PARAMS	CKA_VALUE
Private key	CKO_PRIVATE_KEY	CK_TRUE	CK_TRUE	CK_TRUE or CK_FALSE	CKK_EC	All UTF8 symbols	CK_BYTE[]	CK_BYTE[]
Key Type	**CKA_CLASS**	**CKA_TOKEN**	**CKA_PRIVATE**	**CKA_MODIFIABLE**	**CKA_KEY_TYPE**	**CKA_ID**	**CKA_EC_PARAMS**	**CK_EC_POINT**
Public key	CKO_PUBLIC_KEY	CK_FALSE	CK_ FALSE	CK_TRUE or CK_FALSE	CKK_EC	All UTF8 symbols	CK_BYTE[]	CK_BYTE[]

Table 5: EC key import

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

3.3.2 Generate a keypair

Table 6 shows the attributes necessary to generate a keypair. We only support this feature for token objects.

When calling the function C_GenerateKeyPair(…) the **same template** (the same pointer can be used) for **pPublicKeyTemplate** and **pPublicKeyTemplate** shall be used.

We do not support the generation of keypairs for session objects.

Keypair type	**CKA_CLASS**	**CKA_TOKEN**	**CKA_PRIVATE**	**CKA_KEY_TYPE**	**CKA_MODULUS_BITS**
RSA KeyPair (CKM_RSA_PKCS_KEY_PAIR_GEN)	CKO_PRIVATE_KEY	CK_TRUE	CK_TRUE	CKK_RSA	512 – 2048
Keypair type	**CKA_CLASS**	**CKA_TOKEN**	**CKA_PRIVATE**	**CKA_KEY_TYPE**	**CKA_EC_PARAMS**
EC KeyPair (CKM_ECDSA_KEY_PAIR_GEN)	CKO_PRIVATE_KEY	CK_TRUE	CK_TRUE	CKK_EC	use one OID from Table 4

Table 6: generate a keypair

3.3.3 Change attributes

Currently we do only support to change CKA_ID.

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

4 Differences to standard cryptoki specification

Cryptoki is defined very strictly while some parts are quite vague since tokens may vary in their behavior. In order to support our applet we had to violate some of these definitions. This chapter shows all the differences compared with the original specification.

4.1 KeyGeneration

- Use the same template for pPublicKeyTemplate and pPrivateKeyTemplate
- For EC keypair generation we limit CKA_EC_PARAMS for OID only usage. Supported OIDs can be found in Table 4

4.2 Import

- When importing RSA keypairs do not use CKA_MODULUS_BITS attribute since this is internally used for detecting keypair generation
- When importing RSA keypairs do not import the public key since this object is automatically generated on private key import. Just import the private key and use the object search afterwards to find the corresponding public key. Use the CKA_ID to find it since it will have the same CKA_ID attribute as the private key.
- For EC keypair import we limit CKA_EC_PARAMS for OID only usage. Supported OIDs can be found in Table 4
- When importing EC keypairs do not import the public key since this object is automatically generated on private key import. Just import the private key and use the object search afterwards to find the corresponding public key. Use the CKA_ID to find it since it will have the same CKA_ID attribute as the private key.

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

5 Examples

5.1 Import private key and search associated public key

```
CK_BYTE p[] = { 0xd8, 0x62, 0x45, 0x54, 0x66, 0x02, 0xea, 0x8f, 0xde, 0x9f, 0xff, 0xe1, 0xc4, 0x34, 0xf7, 0x83, 0x1e,
0xd7, 0x9d, 0x2e, 0xf7, 0x66, 0xc6, 0xfd, 0x6e, 0x81, 0xf7, 0xa5, 0x05, 0xed, 0x4d, 0xc6, 0x06, 0xaa, 0x2e, 0xf3, 0x37,
0x78, 0x07, 0x03, 0x46, 0x5a, 0x51, 0xa2, 0x4d, 0x9f, 0xd2, 0x3a, 0x11, 0xff, 0x89, 0x4f, 0x21, 0xff, 0x3d, 0xa8, 0xa7,
0x0a, 0xe8, 0x3a, 0x2e, 0xcd, 0xfa, 0x75, 0xdd, 0x11, 0x3a, 0xf6, 0x29, 0x49, 0x54, 0xd8, 0x06, 0xc9, 0xa0, 0xb1, 0x47,
0x23, 0x5e, 0xc3, 0x52, 0x19, 0xae, 0x72, 0x54, 0x66, 0x71, 0xd9, 0xff, 0x9a, 0x4a, 0x13, 0x6e, 0x44, 0xb7, 0x42, 0x9f,
0x1a, 0xe4, 0xf2, 0xce, 0xe1, 0x94, 0x57, 0x27, 0x3d, 0x9d, 0xd5, 0x19, 0x73, 0xfe, 0x9b, 0x29, 0x50, 0x21, 0xc1, 0xea,
0x8e, 0x71, 0xbe, 0x5f, 0xff, 0xe5, 0x3a, 0xd5, 0xc4, 0x8e, 0x3b };

CK_BYTE q[] = { 0xdd, 0x81, 0x04, 0xb3, 0xd9, 0x47, 0x28, 0xa4, 0xeb, 0x95, 0x70, 0x14, 0x18, 0xa4, 0x11, 0x07, 0xe9,
0xec, 0x10, 0x44, 0xd5, 0x63, 0xa9, 0x52, 0xc9, 0x39, 0x49, 0x43, 0xd8, 0x05, 0xaf, 0xa2, 0x60, 0xcc, 0xe3, 0x49, 0xb4,
0x52, 0x38, 0xd4, 0x71, 0xc1, 0x5b, 0x75, 0x60, 0x46, 0xe1, 0xff, 0x46, 0xce, 0x5c, 0xdf, 0x97, 0xe1, 0x00, 0x89, 0x05,
0x16, 0x80, 0x3d, 0x15, 0x7e, 0x03, 0xbf, 0x09, 0xb6, 0x9e, 0x7e, 0x38, 0x55, 0xed, 0x47, 0x58, 0x9d, 0xc9, 0x72, 0x39,
0xe6, 0x50, 0xc4, 0x93, 0xe5, 0x36, 0x41, 0x81, 0x3e, 0xef, 0xdd, 0x29, 0xab, 0xc5, 0xdd, 0x25, 0x77, 0x85, 0x3c, 0x25,
0xb4, 0x94, 0xfe, 0x4c, 0x0d, 0x43, 0xef, 0x29, 0x61, 0x99, 0xbe, 0xa5, 0x71, 0x65, 0xdd, 0x1f, 0xde, 0xa9, 0x87, 0x6c,
0x85, 0x26, 0x42, 0x7b, 0x49, 0x68, 0xdc, 0x16, 0x5b, 0xa6, 0xeb };

CK_BYTE publicExponent[] = { 0x01, 0x00, 0x01 };

CK_OBJECT_HANDLE publicKeyObjectHandle = 0;
CK_OBJECT_HANDLE privateKeyObjectHandle = 0;

CK_ATTRIBUTE aClass, aKeyType, aID, aToken, aPrivate;
CK_OBJECT_CLASS classType = CKO_PRIVATE_KEY;
CK_KEY_TYPE keyType = CKK_RSA;
CK_BYTE ID[16];

fillRandomID( ID, 16 );

CK_BBOOL trueValue = CK_TRUE;

aClass.type = CKA_CLASS;
aKeyType.type = CKA_KEY_TYPE;
aID.type = CKA_ID;
aToken.type = CKA_TOKEN;
aPrivate.type = CKA_PRIVATE;
```

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

```
aClass.pValue = &classType;
aKeyType.pValue = &keyType;
aID.pValue = ID;
aToken.pValue = &trueValue;
aPrivate.pValue = &trueValue;

aClass.ulValueLen = sizeof( classType );
aKeyType.ulValueLen = sizeof( keyType );
aID.ulValueLen = sizeof( ID );
aToken.ulValueLen = sizeof( trueValue );
aPrivate.ulValueLen = sizeof( trueValue );

CK_ATTRIBUTE aPrimeP, aPrimeQ, aPublicExponent;

aPrimeP.type = CKA_PRIME_1;
aPrimeQ.type = CKA_PRIME_2;
aPublicExponent.type = CKA_PUBLIC_EXPONENT;

aPrimeP.pValue = p.data();
aPrimeQ.pValue = q.data();
aPublicExponent.pValue = publicExponent.data();

aPrimeP.ulValueLen = p.size();
aPrimeQ.ulValueLen = q.size();
aPublicExponent.ulValueLen = publicExponent.size();

CK_ATTRIBUTE attributes[] = {aClass, aKeyType, aID, aToken, aPrivate, aPrimeP, aPrimeQ, aPublicExponent};

pList->C_CreateObject( session, attributes, sizeof( attributes ) / sizeof( CK_ATTRIBUTE ), &privateKeyObjectHandle );

//find corresponding public key
CK_OBJECT_CLASS anotherClassType = CKO_PUBLIC_KEY;
aClass.pValue = &anotherClassType;

CK_ATTRIBUTE publicKeyAttributes[] = {aClass, aID};
CK_ULONG count = 1;

pList->C_FindObjectsInit( session, publicKeyAttributes, sizeof( publicKeyAttributes ) / sizeof( CK_ATTRIBUTE ) );
```

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

```
pList->C_FindObjects( session, &publicKeyObjectHandle, 1, &count );
pList->C_FindObjectsFinal( session );
```

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

5.1.1 Create RSA public key session object to verify some signature

```
CK_OBJECT_HANDLE publicKeyHandle;

CK_BYTE data[] = {'h', 'a', 'l', 'l', 'o'};

CK_BYTE signature[] = {0x8E, 0x99, 0xE3, 0x0D, 0xB4, 0xA6, 0x44, 0x0D, 0x91, 0x9D, 0x64, 0x07, 0xB4, 0xED, 0x07, 0x4F,
0x83, 0x80, 0xA8, 0x41, 0x9E, 0xB0, 0xB4, 0xE3, 0x74, 0x77, 0x48, 0xD7, 0x2F, 0xE4, 0xAC, 0x74, 0xF4, 0xD5, 0xE2, 0x3A,
0xAB, 0xB7, 0xD4, 0xC7, 0xE8, 0xD9, 0x54, 0xD6, 0x61, 0xB9, 0xC9, 0x01, 0xBF, 0xA6, 0x02, 0x98, 0x33, 0xDC, 0xA0, 0x9B,
0x4E, 0xBF, 0xFF, 0xB8, 0x07, 0x02, 0x35, 0x72};

CK_BYTE modulus[] = {0x99, 0x69, 0x9D, 0x45, 0x39, 0xDB, 0x53, 0x68, 0xC4, 0x6D, 0xEB, 0x6E, 0x4C, 0x25, 0xEC, 0x5B,
0x84, 0xBE, 0x17, 0xC8, 0x85, 0xF5, 0x63, 0x03, 0x2E, 0x3A, 0xFE, 0x59, 0x92, 0x9A, 0x29, 0x3C, 0xD0, 0x4F, 0x7B, 0x57,
0xB7, 0x11, 0x9D, 0x5E, 0xE6, 0x86, 0x5B, 0x03, 0xA0, 0xD9, 0xA9, 0xEA, 0x2E, 0xB9, 0x15, 0xA7, 0xBF, 0x80, 0xA6, 0x69,
0x6E, 0x23, 0x5A, 0x01, 0xDE, 0xCE, 0xED, 0x5D};

CK_BYTE public_exponent[] = {0x01, 0x00, 0x01};

CK_ATTRIBUTE aClass, aKeyType, aID, aToken, aPrivate;
CK_OBJECT_CLASS classType = CKO_PUBLIC_KEY;
CK_KEY_TYPE keyType = CKK_RSA;
CK_BYTE ID[] = {'t', 'e', 's', 't', '_', 'R', 'S', 'A', '_', '5', '1', '2'};
CK_BBOOL trueValue = CK_TRUE;
CK_BBOOL falseValue = CK_FALSE;

aClass.type = CKA_CLASS;
aKeyType.type = CKA_KEY_TYPE;
aID.type = CKA_ID;
aToken.type = CKA_TOKEN;
aPrivate.type = CKA_PRIVATE;

aClass.pValue = &classType;
aKeyType.pValue = &keyType;
aID.pValue = ID;
aToken.pValue = &falseValue;
aPrivate.pValue = &falseValue;

aClass.ulValueLen = sizeof( classType );
aKeyType.ulValueLen = sizeof( keyType );
```

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

```
aID.ulValueLen = sizeof( ID );
aToken.ulValueLen = sizeof( falseValue );
aPrivate.ulValueLen = sizeof( falseValue );

CK_ATTRIBUTE aModulus, aPublicExponent;

aModulus.type = CKA_MODULUS;
aPublicExponent.type = CKA_PUBLIC_EXPONENT;

aModulus.pValue = modulus;
aPublicExponent.pValue = public_exponent;

aModulus.ulValueLen = sizeof( modulus );
aPublicExponent.ulValueLen = sizeof( public_exponent );

CK_ATTRIBUTE attributes[] = {aClass, aKeyType, aID, aToken, aPrivate, aModulus, aPublicExponent};
pList->C_CreateObject( sessionID, attributes, sizeof( attributes ) / sizeof( CK_ATTRIBUTE ), &publicKeyHandle );

CK_MECHANISM mechanism;
mechanism.mechanism = CKM_SHA1_RSA_PKCS;

pList->C_VerifyInit( sessionID, &mechanism, publicKeyHandle);
pList->C_Verify( sessionID, data, sizeof( data ), signature, sizeof( signature ));
```

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

5.1.2 Generate EC keypair

```
CK_OBJECT_HANDLE publicKeyHandle, privateKeyHandle;
CK_MECHANISM mech;
mech.mechanism = CKM_ECDSA_KEY_PAIR_GEN;

CK_ATTRIBUTE aClass, aToken, aPrivate, aKeyType, aECParams;

CK_OBJECT_CLASS classType = CKO_PRIVATE_KEY;
CK_BBOOL falseValue = CK_FALSE;
CK_BBOOL trueValue = CK_TRUE;
CK_KEY_TYPE keyType = CKK_EC;

aClass.type = CKA_CLASS;
aClass.pValue = &classType;
aClass.ulValueLen = sizeof( classType );

aToken.type = CKA_TOKEN;
aToken.pValue = &trueValue;
aToken.ulValueLen = sizeof( trueValue );

aPrivate.type = CKA_PRIVATE;
aPrivate.pValue = &trueValue;
aPrivate.ulValueLen = sizeof( trueValue );

aKeyType.type = CKA_KEY_TYPE;
aKeyType.pValue = &keyType;
aKeyType.ulValueLen = sizeof( keyType );

aECParams.type = CKA_EC_PARAMS;
aECParams.pValue = oid;
aECParams.ulValueLen = oidSize;

CK_ATTRIBUTE attributes[] = {aClass, aToken, aPrivate, aKeyType, aECParams};

pList->C_GenerateKeyPair( session, &mech, attributes, sizeof( attributes ) / sizeof( CK_ATTRIBUTE ), attributes, sizeof(
attributes ) / sizeof( CK_ATTRIBUTE ), &publicKeyHandle, &privateKeyHandle );
```

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

5.1.3 Change attribute

```
//... privateKey & publicKey have been created / generated previously -> we change their IDs and Label

CK_ATTRIBUTE aID, aLabel;
char test[] = "Test";

char ID[4];
char LABEL[4];

aID.type = CKA_ID;
aID.pValue = test;
aID.ulValueLen = 4;

aLabel.type = CKA_LABEL;
aLabel.pValue = test;
aLabel.ulValueLen = 4;

CK_ATTRIBUTE attributes[] = {aID, aLabel};

pFunctionList->C_SetAttributeValue( session, privateKey, attributes, sizeof( attributes ) / sizeof( CK_ATTRIBUTE ) );

pFunctionList->C_SetAttributeValue( session, publicKey, attributes, sizeof( attributes ) / sizeof( CK_ATTRIBUTE ) );
```

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.

6 References

[CRTK04] *RSA Laboratories, 2004, PKCS #11 v2.20: Cryptographic Token Interface Standard*

© certgate 2016 - All rights reserved.
No part of this documentation may be used for any purposes other than personal use, or processed, duplicated or distributed using electronic systems, in any form such as print, photocopy or any other procedure without a prior written approval from certgate GmbH.