Pycryptoki Documentation

Release 2.5.0

Gemalto

Jul 25, 2019

Contents

CHAPTER 1

Overview

Pycryptoki is an open-source Python wrapper around Safenet's C PKCS11 library. Using python's ctypes library, we can simplify memory management, and provide easy, pythonic access to a PKCS11 shared library.

The primary function of pycryptoki is to *simplify* PKCS11 calls. Rather than needing to calculate data sizes, buffers, or other low-level memory manipulation, you simply need to pass in data.

It's highly recommended that you have the PKCS11 documentation handy, as pycryptoki uses that as the underlying C interface. Session management, object management, and other concepts are unchanged from PKCS11.

```
from pycryptoki.default_templates import *
from pycryptoki.defines import *
from pycryptoki.key_generator import *
from pycryptoki.session_management import *


c_initialize_ex()
auth_session = c_open_session_ex(0)    # HSM slot # in this example is 0
login_ex(auth_session, 0, 'userpin')  # 0 is still the slot number, 'userpin' should␣
↪be replaced by your password (None if PED or no challenge)

# Get some default templates
# They are simple python dictionaries, and can be modified to suit needs.
pub_template, priv_template = get_default_key_pair_template(CKM_RSA_PKCS_KEY_PAIR_GEN)

# Modifying template would look like:
pub_template[CKA_LABEL] = b"RSA PKCS Pub Key"
pub_template[CKA_MODULUS_BITS] = 2048   # 2048 key size

pubkey, privkey = c_generate_key_pair_ex(auth_session, CKM_RSA_PKCS_KEY_PAIR_GEN, pub_
↪template, priv_template)
print("Generated Private key at %s and Public key at %s" % (privkey, pubkey))

c_logout_ex(auth_session)
c_close_session_ex(auth_session)
c_finalize_ex()
```

1.1 Getting Started

To use pycryptoki, you must have SafeNet LunaClient installed.

1.1.1 Installation

Pycryptoki can be installed on any machine that has Python installed. Python versions >= 2.7 are supported.:

```
pip install git+https://github.com/gemalto/pycryptoki
```

Pycryptoki will attempt to auto-locate the SafeNet Cryptoki shared library when pycryptoki is first called. It will use the configuration files as defined by the LunaClient documentation to determine which library to use.

1.1.2 Simple Example

This example will print out information about the given token slot.

```
from pycryptoki.session_management import (c_initialize_ex,
                                           c_get_info_ex,
                                           get_firmware_version,
                                           c_get_token_info_ex,
                                           c_finalize_ex)


c_initialize_ex()
print("C_GetInfo: ")
print("\n".join("\t{}: {}".format(x, y) for x, y in c_get_info_ex().items()))
token_info = c_get_token_info_ex(0)
print("C_GetTokenInfo:")
print("\n".join("\t{}: {}".format(x, y) for x, y in token_info.items()))
print("Firmware version: {}".format(get_firmware_version(0)))

c_finalize_ex()
```

1.2 Examples

1.2.1 Generating an RSA Key Pair

This example creates a 1024b RSA Key Pair.

```
from pycryptoki.session_management import (c_initialize_ex, c_finalize_ex,
                                           c_open_session_ex, c_close_
↪session_ex,
                                           login_ex)
from pycryptoki.defines import CKM_RSA_PKCS_KEY_PAIR_GEN
from pycryptoki.key_generator import c_generate_key_pair_ex

c_initialize_ex()
session = c_open_session_ex(0)       # 0 -> slot number
login_ex(session, 0, 'userpin')      # 0 -> Slot number, 'userpin' -> token␣
↪password
```

(continues on next page)

(continued from previous page)

```
# Templates are dictionaries in pycryptoki
pub_template = {CKA_TOKEN: True,
                CKA_PRIVATE: True,
                CKA_MODIFIABLE: True,
                CKA_ENCRYPT: True,
                CKA_VERIFY: True,
                CKA_WRAP: True,
                CKA_MODULUS_BITS: 1024,  # long 0 - MAX_RSA_KEY_NBITS
                CKA_PUBLIC_EXPONENT: 3,  # byte
                CKA_LABEL: b"RSA Public Key"}
priv_template = {CKA_TOKEN: True,
                 CKA_PRIVATE: True,
                 CKA_SENSITIVE: True,
                 CKA_MODIFIABLE: True,
                 CKA_EXTRACTABLE: True,
                 CKA_DECRYPT: True,
                 CKA_SIGN: True,
                 CKA_UNWRAP: True,
                 CKA_LABEL: b"RSA Private Key"}

pub_key, priv_key = c_generate_key_pair_ex(session,
                                           mechanism=CKM_RSA_PKCS_KEY_PAIR_
↪GEN,
                                           pbkey_template=pub_template,
                                           prkey_template=priv_template)

c_close_session_ex(session)
c_finalize_ex()
```

1.2.2 Encrypting data with AES-CBC-PAD

This example generates a 24-byte AES key, then encrypts some data with that key using the AES-CBC-PAD mechanism.

```
from pycryptoki.session_management import (c_initialize_ex, c_finalize_ex,
                                           c_open_session_ex, c_close_
↪session_ex,
                                           login_ex)
from pycryptoki.defines import (CKM_AES_KEY_GEN,
                                CKA_LABEL,
                                CKA_ENCRYPT,
                                CKA_DECRYPT,
                                CKA_TOKEN,
                                CKA_CLASS,
                                CKA_KEY_TYPE,
                                CKK_AES,
                                CKO_SECRET_KEY,
                                CKA_SENSITIVE,
                                CKA_WRAP,
                                CKA_UNWRAP,
                                CKA_DERIVE,
                                CKA_VALUE_LEN,
                                CKA_EXTRACTABLE,
                                CKA_PRIVATE,
```

(continues on next page)

(continued from previous page)

```
                                    CKM_AES_CBC_PAD)
from pycryptoki.key_generator import c_generate_key_ex
from pycryptoki.encryption import c_encrypt_ex
from pycryptoki.conversions import to_bytestring, from_hex
from pycryptoki.mechanism import Mechanism

c_initialize_ex()
session = c_open_session_ex(0)      # 0 = slot number
login_ex(session, 0, 'userpin')        # 'userpin' = token password


template = {CKA_LABEL: b"Sample AES Key",
            CKA_ENCRYPT: True,
            CKA_DECRYPT: True,
            CKA_TOKEN: False,
            CKA_CLASS: CKO_SECRET_KEY,
            CKA_KEY_TYPE: CKK_AES,
            CKA_SENSITIVE: True,
            CKA_PRIVATE: True,
            CKA_WRAP: True,
            CKA_UNWRAP: True,
            CKA_DERIVE: True,
            CKA_VALUE_LEN: 24,
            CKA_EXTRACTABLE: True,}
aes_key = c_generate_key_ex(session, CKM_AES_KEY_GEN, template)

# Data is in hex format here
raw_data = "d0d77c63ab61e75a5fd4719fa77cc2de1d817efedcbd43e7663736007672e8c7"

# Convert to raw bytes before passing into c_encrypt:
data_to_encrypt = to_bytestring(from_hex(raw_data))


# Note: this is *bad crypto practice*! DO NOT USE STATIC IVS!!
mechanism = Mechanism(mech_type=CKM_AES_CBC_PAD,
                      params={"iv": list(range(16))})
static_iv_encrypted_data = c_encrypt_ex(session, aes_key, data_to_encrypt,␣
↪mechanism)

c_close_session_ex(session)
c_finalize_ex()
```

1.2.3 Finding a key and decrypting Data

This example follows from the previous one, except instead of generating a key, we'll find one that was already used.

```
from pycryptoki.session_management import (c_initialize_ex, c_finalize_ex,
                                           c_open_session_ex, c_close_session_ex,
                                           login_ex)
from pycryptoki.object_attr_lookup import c_find_objects_ex
from pycryptoki.defines import (CKM_AES_KEY_GEN,
                                CKA_LABEL,
                                CKA_ENCRYPT,
                                CKA_DECRYPT,
                                CKA_TOKEN,
```

(continues on next page)

(continued from previous page)

```
                                CKA_CLASS,
                                CKA_KEY_TYPE,
                                CKK_AES,
                                CKO_SECRET_KEY,
                                CKA_SENSITIVE,
                                CKA_WRAP,
                                CKA_UNWRAP,
                                CKA_DERIVE,
                                CKA_VALUE_LEN,
                                CKA_EXTRACTABLE,
                                CKA_PRIVATE,
                                CKM_AES_CBC_PAD)
from pycryptoki.encryption import c_decrypt_ex
from pycryptoki.conversions import to_bytestring, from_hex
from pycryptoki.mechanism import Mechanism

c_initialize_ex()
session = c_open_session_ex(0)       # 0 = slot number
login_ex(session, 0, 'userpin')         # 'userpin' = token password

template = {CKA_LABEL: b"Sample AES key"}

keys = c_find_objects_ex(session, template, 1)
aes_key = keys.pop(0) # Use the first key found.

# Data is in hex format here
raw_data =
↪"95e28bc6da451f3064d688dd283c5c43a5dd374cb21064df836e2970e1024c2448f129062aacbae3e45abd098b893346
↪"

# Convert to raw bytes before passing into c_decrypt:
data_to_decrypt = to_bytestring(from_hex(raw_data))


# Note: this is *bad crypto practice*! DO NOT USE STATIC IVS!!
mechanism = Mechanism(mech_type=CKM_AES_CBC_PAD,
                      params={"iv": list(range(16))})
original_data = c_decrypt_ex(session, aes_key, data_to_decrypt, mechanism)

c_close_session_ex(session)
c_finalize_ex()
```

1.3 Frequent Issues

Contents

1.3.1 Wrong data type

Any cryptographic function working on data (ex. `c_encrypt`, `c_unwrap`) will expect a bytestring. A string object in Python2 is by default a *bytestring*, but in Python3 is a *unicode* string.

For example:

```
c_encrypt(session, key, "this is some test data", mechanism)
```

Will work in Python 2, but NOT Python 3. Instead, use the *pycryptoki.conversions* module to ensure that any data you pass into the cryptoki library is of the correct form.

Another 'gotcha' is that hex data represented as a string that is then used in an encrypt call would result in 2x the length of expected data:

```
from pycryptoki.conversions import to_bytestring, from_hex
hex_data = "deadbeef"
assert len(hex_data) == 8
raw_data = list(from_hex(hex_data))
assert len(raw_data) == 4
print (raw_data)
# Prints: [222, 173, 190, 239]
```

Another example:

```
from pycryptoki.conversions import to_bytestring, from_hex
some_hex_data = "06abde23df89"
data_to_encrypt = to_bytestring(from_hex(some_hex_data))
c_encrypt(session, key, data_to_encrypt, mechanism)
```

Note:

See this article for more details about the differences between unicode and bytestrings in python: http://lucumr.pocoo.org/2014/1/5/unicode-in-2-and-3/

1.3.2 Internal Initialization Vectors

When you use an internal IV for AES mechanisms, the IV is appended to the cipher text. This needs to be stripped off and used to create the mechanism for decryption:

```
from pycryptoki.encryption import c_encrypt_ex

data_to_encrypt = b"a" * 64
mech = Mechanism(CKM_AES_KW,
                 params={"iv": []}) # Uses an internal IV

enc_data = c_encrypt_ex(session, key, data_to_encrypt, mech)
iv = enc_data[-16:] # Strip off the last 16 bytes of the encrypted data.
decrypt_mech = Mechanism(CKM_AES_KW,
                         params={"iv": iv})
decrypted_data = c_decrypt_ex(session, key, enc_data[:-16], decrypt_mech)
```

1.3.3 PKCS11 Calling Conventions

The PKCS11 library has two main methods for returning data to the caller:

1. Allocate a large enough buffer for the resulting data and make the PKCS11 call with that buffer.
2. Call the function with a NULL pointer for the buffer. The PKCS11 library will then place the required buffer size in `*pulBufLen`.

Pycryptoki will let you perform either method for any function that returns data in a variable-length buffer with the `output_buffer` keyword argument. This argument takes either an integer, or a list of integers. The integer specifies the *size* of the buffer to use for the returned output. This means if you use a very small integer, you could get back `CKR_BUFFER_TOO_SMALL` (and you could also allocate a buffer that is incredibly large – limited by the memory of your system).

By default, pycryptoki will use method #2 (querying the library for buffer size):

```
data = b"deadbeef"
c_decrypt_ex(session, key, data, mechanism)
```

Will result in the raw underlying PKCS11 calls:

```
DEBUG: Cryptoki call: C_DecryptInit(8, <pycryptoki.cryptoki.CK_MECHANISM object at␣
↪0x7f693480c598>, c_ulong(26))
DEBUG: Cryptoki call: C_Decrypt(8, <pycryptoki.cryptoki.LP_c_ubyte object at␣
↪0x7f69347df598>, c_ulong(2056), None, <pycryptoki.cryptoki.LP_c_ulong object at␣
↪0x7f69347dfbf8>)
DEBUG: Allocating <class 'ctypes.c_ubyte'> buffer of size: 2048
DEBUG: Cryptoki call: C_Decrypt(8, <pycryptoki.cryptoki.LP_c_ubyte object at␣
↪0x7f69347df598>, c_ulong(2056), <pycryptoki.cryptoki.LP_c_ubyte object at␣
↪0x7f693498c9d8>, <pycryptoki.cryptoki.LP_c_ulong object at 0x7f693498c840>)
```

Note: `None` in python is the equivalent to `NULL` in C.

An example using a pre-allocated buffer:

```
data = b"deadbeef"
c_decrypt_ex(session, key, data, mechanism, output_buffer=0xffff)
```

And the resulting PKCS11 calls:

```
DEBUG: Cryptoki call: C_DecryptInit(8, <pycryptoki.cryptoki.CK_MECHANISM object at␣
↪0x7f693480c598>, c_ulong(26))
DEBUG: Allocating <class 'ctypes.c_ubyte'> buffer of size: 2048
DEBUG: Cryptoki call: C_Decrypt(8, <pycryptoki.cryptoki.LP_c_ubyte object at␣
↪0x7f69347df598>, c_ulong(2056), <pycryptoki.cryptoki.LP_c_ubyte object at␣
↪0x7f693498c9d8>, <pycryptoki.cryptoki.LP_c_ulong object at 0x7f693498c840>)
```

For multi-part operations, `output_buffer` should be a list of integers of equal size to the number of parts in the operation:

```
data = [b"a" * 8, b"b" * 8, b"c" * 8, b"d" * 8]
output_buffer = [0xffff] * len(data)  # Equivalent to: [0xffff, 0xffff, 0xffff,␣
↪0xffff]
c_encrypt_ex(session, key, data, mechanism, output_buffer=output_buffer)
```

For a multi-part operation that returns data in the `C_*Final` function, the output buffer will be equivalent to the largest buffer size specified in the output_buffer list.

1.4 API Reference

There are some general guidelines to using pycryptoki:

1. If you want to perform a PKCS11 operation as a multi-part operation, provide the input data as a list or a tuple.
2. Data should always be passed into `c_` functions as raw byte data (bytestrings). Conversions are available to convert hex data or binary data to bytes at *pycryptoki.conversions*
3. Returned encrypted/decrypted data is always raw bytestrings.

1.4.1 Session/Token Management

Modules for Token and session creation and management.

Contents

1.4.1.1 Session Management

Methods responsible for managing a user's session and login/c_logout

pycryptoki.session_management.**c_initialize**(*flags=None*, *init_struct=None*)

Initializes current process for use with PKCS11.

Some sample flags:

CKF_LIBRARY_CANT_CREATE_OS_THREADS CKF_OS_LOCKING_OK

See the PKCS11 documentation for more details.

Parameters

- **flags** (*int*) – Flags to be set within InitArgs Struct. (Default = None)
- **init_struct** – InitArgs structure (Default = None)

Returns Cryptoki return code.

pycryptoki.session_management.**c_initialize_ex**(*flags=None*, *init_struct=None*)

Executes *c_initialize()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_finalize**()
Finalizes PKCS11 library.

Returns Cryptoki return code

pycryptoki.session_management.**c_finalize_ex**()
Executes *c_finalize()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_open_session**(*slot_num*, *flags=6*)
Opens a session on the given slot

Parameters

- **slot_num** (*int*) – The slot to get a session on
- **flags** (*int*) – The flags to open the session with (Default value = (CKF_SERIAL_SESSION | CKF_RW_SESSION)

Returns (retcode, session handle)

Return type tuple

pycryptoki.session_management.**c_open_session_ex**(*slot_num*, *flags=6*)
Executes *c_open_session()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**login**(*h_session*, *slot_num=1*, *password=None*, *user_type=1*)
Login to the given session.

Parameters

- **h_session** (*int*) – Session handle
- **slot_num** (*int*) – Slot index to login on (Default value = 1)
- **password** (*bytes*) – Password to login with (Default value = "userpin")
- **user_type** (*int*) – User type to login as (Default value = 1)

Returns retcode

Return type int

pycryptoki.session_management.**login_ex**(*h_session*, *slot_num=1*, *password=None*, *user_type=1*)
Executes *login()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_get_info**()
Get general information about the Cryptoki Library

Returns a dictionary containing the following keys:

- cryptokiVersion
- manufacturerID
- flags
- libraryDescription
- libraryVersion

`cryptokiVersion` and `libraryVersion` are *CK_VERSION* structs, and the major/minor values can be accessed directly (`info['cryptokiVersion'].major == 2`)

Returns (retcode, info dictionary)

pycryptoki.session_management.**c_get_info_ex**()
Executes *c_get_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_get_slot_list**(*token_present=True*)
Get a list of all slots.

Parameters **token_present** (*bool*) – If true, will only return slots that have a token present.

Returns List of slots

pycryptoki.session_management.**c_get_slot_list_ex**(*token_present=True*)
Executes *c_get_slot_list()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_get_slot_info**(*slot*)
Get information about the given slot number.

Parameters **slot** (*int*) – Target slot

Returns Dictionary of slot information

pycryptoki.session_management.**c_get_slot_info_ex**(*slot*)
Executes *c_get_slot_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_get_session_info**(*session*)
Get information about the given session.

Parameters **session** (*int*) – session handle

Returns (retcode, dictionary of session information)

Return type tuple

pycryptoki.session_management.**c_get_session_info_ex**(*session*)
Executes *c_get_session_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_get_token_info**(*slot_id*, *rstrip=True*)
Gets the token info for a given slot id

Parameters

- **slot_id** (*int*) – Token slot ID
- **rstrip** (*bool*) – If true, will strip trailing whitespace from char data.

Returns (retcode, A python dictionary representing the token info)

Return type tuple

pycryptoki.session_management.**c_get_token_info_ex**(*slot_id*, *rstrip=True*)
Executes *c_get_token_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**get_slot_dict**(*token_present=False*)
Compiles a dictionary of the available slots

Returns A python dictionary of the available slots

pycryptoki.session_management.**get_slot_dict_ex**(*token_present=False*)
Executes *get_slot_dict()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_close_session**(*h_session*)
Closes a session

Parameters **h_session** (*int*) – Session handle

Returns retcode

Return type int

pycryptoki.session_management.**c_close_session_ex**(*h_session*)
Executes *c_close_session()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_logout**(*h_session*)
Logs out of a given session

Parameters **h_session** (*int*) – Session handle

Returns retcode

Return type int

pycryptoki.session_management.**c_logout_ex**(*h_session*)
Executes *c_logout()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.session_management.**c_init_pin**(*h_session*, *pin*)

Initializes the PIN

Parameters

- **h_session** (*int*) – Session handle
- **pin** – pin to c_initialize

Returns THe result code

pycryptoki.session_management.**c_init_pin_ex**(*h_session*, *pin*)

Executes *c_init_pin()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.session_management.**ca_factory_reset**(*slot*)

Does a factory reset on a given slot

Parameters **slot** – The slot to do a factory reset on

Returns The result code

pycryptoki.session_management.**ca_factory_reset_ex**(*slot*)

Executes *ca_factory_reset()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_set_pin**(*h_session*, *old_pass*, *new_pass*)
Allows a user to change their PIN

Parameters

- **h_session** (*int*) – Session handle
- **old_pass** – The user's old password
- **new_pass** – The user's desired new password

Returns The result code

pycryptoki.session_management.**c_set_pin_ex**(*h_session*, *old_pass*, *new_pass*)
Executes *c_set_pin()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**c_close_all_sessions**(*slot*)
Closes all the sessions on a given slot

Parameters **slot** – The slot to close all sessions on

Returns retcode

Return type int

pycryptoki.session_management.**c_close_all_sessions_ex**(*slot*)
Executes *c_close_all_sessions()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**ca_openapplicationID**(*slot*, *id_high*, *id_low*)
Open an application ID on the given slot.

Parameters

- **slot** (*int*) – Slot on which to open the APP ID
- **id_high** (*int*) – High value of App ID
- **id_low** (*int*) – Low value of App ID

Returns retcode

Return type int

pycryptoki.session_management.**ca_openapplicationID_ex**(*slot*, *id_high*, *id_low*)
Executes *ca_openapplicationID()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**ca_closeapplicationID**(*slot*, *id_high*, *id_low*)
Close a given AppID on a slot.

Parameters

- **slot** (*int*) – Slot on which to close the APP ID
- **id_high** (*int*) – High value of App ID
- **id_low** (*int*) – Low value of App ID

Returns retcode

Return type int

pycryptoki.session_management.**ca_closeapplicationID_ex**(*slot*, *id_high*, *id_low*)
Executes *ca_closeapplicationID()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**ca_setapplicationID**(*id_high*, *id_low*)
Set the App ID for the current process.

Parameters

- **id_high** (*int*) – High value of App ID
- **id_low** (*int*) – Low value of App ID

Returns retcode

Return type int

pycryptoki.session_management.**ca_setapplicationID_ex**(*id_high*, *id_low*)
Executes *ca_setapplicationID()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**ca_restart**(*slot*)

Parameters **slot** –

pycryptoki.session_management.**ca_restart_ex**(*slot*)
Executes *ca_restart()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.session_management.**get_firmware_version**(*slot*)

Returns a string representing the firmware version of the given slot.

It will first try to call CA_GetFirmwareVersion, and if that fails (not present on older cryptoki libraries), will call C_GetTokenInfo.

Parameters **slot** (*int*) – Token slot number

Returns Firmware String in the format "X.Y.Z", where X is major, Y is minor, Z is subminor.

Return type str

1.4.1.2 Token Management

Created on Aug 24, 2012

@author: mhughes

pycryptoki.token_management.**c_init_token**(*slot_num*, *password*, *token_label='Main Token'*)

Initializes at token at a given slot with the proper password and label

Parameters

- **slot_num** – The index of the slot to c_initialize a token in
- **password** – The password to c_initialize the slot with
- **token_label** – The label to c_initialize the slot with (Default value = 'Main Token')

Returns The result code

pycryptoki.token_management.**c_init_token_ex**(*slot_num*, *password*, *token_label='Main Token'*)

Executes *c_init_token()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.token_management.**get_token_by_label**(*label*)

Iterates through all the tokens and returns the first token that has a label that is identical to the one that is passed in

Parameters **label** – The label of the token to search for

Returns The result code, The slot of the token

pycryptoki.token_management.**get_token_by_label_ex**(*label*)
Executes *get_token_by_label()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.token_management.**c_get_mechanism_list**(*slot*)
Gets the list of mechanisms from the HSM

Parameters **slot** – The slot number to get the mechanism list on

Returns The result code, A python dictionary representing the mechanism list

pycryptoki.token_management.**c_get_mechanism_list_ex**(*slot*)
Executes *c_get_mechanism_list()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.token_management.**c_get_mechanism_info**(*slot*, *mechanism_type*)
Gets a mechanism's info

Parameters

- **slot** – The slot to query
- **mechanism_type** – The type of the mechanism to get the information for

Returns The result code, The mechanism info

pycryptoki.token_management.**c_get_mechanism_info_ex**(*slot*, *mechanism_type*)
Executes *c_get_mechanism_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.token_management.**ca_get_token_policies**(*slot*)

Get the policies of the given slot.

Parameters **slot** (*int*) – Target slot number

Returns retcode, {id: val} dict of policies (None if command failed)

pycryptoki.token_management.**ca_get_token_policies_ex**(*slot*)

Executes *ca_get_token_policies()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.2 Key Generation and Management

Contents

1.4.2.1 Key Generation

Methods used to generate keys.

pycryptoki.key_generator.**c_copy_object**(*h_session*, *h_object*, *template=None*)

Method to call the C_CopyObject cryptoki command.

Parameters

- **h_session** (*int*) – Session handle
- **h_object** (*int*) – Handle to the object to be cloned
- **template** (*dict*) – Template for the new object. Defaults to None

Returns (retcode, Handle to the new cloned object)

Return type tuple

pycryptoki.key_generator.**c_copy_object_ex**(*h_session*, *h_object*, *template=None*)

Executes *c_copy_object()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.key_generator.**c_derive_key**(*h_session*, *h_base_key*, *template*, *mechanism=None*)

Derives a key from another key.

Parameters

- **h_session** (*int*) – Session handle
- **h_base_key** (*int*) – The base key
- **template** (*dict*) – A python template of attributes to set on derived key
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns The result code, The derived key's handle

pycryptoki.key_generator.**c_derive_key_ex**(*h_session*, *h_base_key*, *template*, *mechanism=None*)

Executes *c_derive_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.key_generator.**c_destroy_object**(*h_session*, *h_object_value*)
Deletes the object corresponsing to the passed in object handle

Parameters

- **h_session** (*int*) – Session handle
- **h_object_value** (*int*) – The handle of the object to delete

Returns Return code

pycryptoki.key_generator.**c_destroy_object_ex**(*h_session*, *h_object_value*)
Executes *c_destroy_object()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.key_generator.**c_generate_key**(*h_session*, *mechanism=None*, *template=None*)
Generates a symmetric key of a given flavor given the correct template.

Parameters

- **h_session** (*int*) – Session handle
- **template** (*dict*) – The template to use to generate the key
- **mechanism** – See the parse_mechanism() function for possible values.

Returns (retcode, generated key handle)

Rtype tuple

pycryptoki.key_generator.**c_generate_key_ex**(*h_session*, *mechanism=None*, *template=None*)
Executes *c_generate_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.key_generator.**c_generate_key_pair**(*h_session*, *mechanism=None*, *pbkey_template=None*, *prkey_template=None*)

Generates a private and public key pair for a given flavor, and given public and private key templates. The return value will be the handle for the key.

Parameters

- **h_session** (*int*) – Session handle
- **pbkey_template** (*dict*) – The public key template to use for key generation
- **prkey_template** (*dict*) – The private key template to use for key generation
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns (retcode, public key handle, private key handle)

Return type tuple

pycryptoki.key_generator.**c_generate_key_pair_ex**(*h_session*, *mechanism=None*, *pbkey_template=None*, *prkey_template=None*)

Executes *c_generate_key_pair()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.key_generator.**clear_keys**(*h_session*)

Quick hacked together function that can be used to clear the first 10 000 keys.

Parameters **h_session** (*int*) – Session handle

1.4.2.2 Key Management

Methods responsible for key management

pycryptoki.key_management.**ca_generatemofn**(*h_session*, *m_value*, *vector_value*, *vector_count*, *is_secure_port_used*)

Generates MofN secret information on a token.

Parameters

- **h_session** (*int*) – Session handle
- **m_value** – m
- **vector_count** – number of vectors

- **is_secure_port_used** – is secure port used
- **vector_value** –

Returns the result code

pycryptoki.key_management.**ca_generatemofn_ex**(*h_session*, *m_value*, *vector_value*, *vector_count*, *is_secure_port_used*)

Executes *ca_generatemofn()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.key_management.**ca_modifyusagecount**(*h_session*, *h_object*, *command_type*, *value*)

Modifies CKA_USAGE_COUNT attribute of the object.

Parameters

- **h_session** (*int*) – Session handle
- **h_object** – object
- **command_type** – command type
- **value** – value

Returns the result code

pycryptoki.key_management.**ca_modifyusagecount_ex**(*h_session*, *h_object*, *command_type*, *value*)

Executes *ca_modifyusagecount()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.2.3 Key Usage

Methods responsible for key usage

pycryptoki.key_usage.**ca_clonemofn**(*h_session*)
Clones MofN secret from one token to another.

Parameters **h_session** (*int*) – Session handle

Returns the result code

pycryptoki.key_usage.**ca_clonemofn_ex**(*h_session*)
Executes *ca_clonemofn()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.key_usage.**ca_duplicatemofn**(*h_session*)
Duplicates a set of M of N vectors.

Parameters **h_session** (*int*) – Session handle

Returns the result code

pycryptoki.key_usage.**ca_duplicatemofn_ex**(*h_session*)
Executes *ca_duplicatemofn()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.3 Encryption/Decryption

Contents

1.4.3.1 Encryption

pycryptoki.encryption.**c_encrypt**(*h_session*, *h_key*, *data*, *mechanism*, *output_buffer=None*)
Encrypts data with a given key and encryption flavor encryption flavors

---

Note: If data is a list or tuple of strings, multi-part encryption will be used.

Parameters

- **h_session** (*int*) – Current session
- **h_key** (*int*) – The key handle to encrypt the data with
- **data** – The data to encrypt, either a bytestring or a list of bytestrings. If this is a list a multipart operation will be used

  ---

  Note: This will be converted to hexadecimal by calling:

  ```
  to_hex(from_bytestring(data))
  ```

  If you need to pass in raw hex data, call:

  ```
  to_bytestring(from_hex(hex-data))
  ```

  References:

  - *to_hex()*
  - *from_hex()*
  - *to_bytestring()*
  - *from_bytestring()*

- **mechanism** – See the `parse_mechanism()` function for possible values.
- **output_buffer** (*list | int*) – Integer or list of integers that specify a size of output buffer to use for an operation. By default will query with NULL pointer buffer to get required size of buffer.

Returns (Retcode, Python bytestring of encrypted data)

Return type tuple

pycryptoki.encryption.**c_encrypt_ex**(*h_session*, *h_key*, *data*, *mechanism*, *output_buffer=None*)
Executes *c_encrypt()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.3.2 Decryption

pycryptoki.encryption.**c_decrypt**(*h_session*, *h_key*, *encrypted_data*, *mechanism*, *output_buffer=None*)
Decrypt given data with the given key and mechanism.

Note: If data is a list or tuple of strings, multi-part decryption will be used.

Parameters

- **h_session** (*int*) – The session to use
- **h_key** (*int*) – The handle of the key to use to decrypt
- **encrypted_data** (*bytes*) – Data to be decrypted

Note: Data will be converted to hexadecimal by calling:

```
to_hex(from_bytestring(data))
```

If you need to pass in raw hex data, call:

```
to_bytestring(from_hex(hex-data))
```

References:

- *to_hex()*
- *from_hex()*
- *to_bytestring()*
- *from_bytestring()*

- **mechanism** – See the `parse_mechanism()` function for possible values.

- **output_buffer** (*list* / *int*) – Integer or list of integers that specify a size of output buffer to use for an operation. By default will query with NULL pointer buffer to get required size of buffer.

Returns (Retcode, Python bytestring of decrypted data))

Return type tuple

pycryptoki.encryption.**c_decrypt_ex**(*h_session*, *h_key*, *encrypted_data*, *mechanism*, *output_buffer=None*)

Executes *c_decrypt()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.3.3 Key Wrapping/Unwrapping

pycryptoki.encryption.**c_wrap_key**(*h_session*, *h_wrapping_key*, *h_key*, *mechanism*, *output_buffer=None*)

Wrap a key off the HSM into an encrypted data blob.

Parameters

- **h_session** (*int*) – The session to use
- **h_wrapping_key** (*int*) – The handle of the key to use to wrap another key
- **h_key** (*int*) – The key to wrap based on the encryption flavor
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns (Retcode, python bytestring representing wrapped key)

Return type tuple

pycryptoki.encryption.**c_wrap_key_ex**(*h_session*, *h_wrapping_key*, *h_key*, *mechanism*, *output_buffer=None*)

Executes *c_wrap_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.encryption.**c_unwrap_key**(*h_session*, *h_unwrapping_key*, *wrapped_key*, *key_template*, *mechanism*)

Unwrap a key from an encrypted data blob.

Parameters

- **h_session** (*int*) – The session to use
- **h_unwrapping_key** (*int*) – The wrapping key handle
- **wrapped_key** (*bytes*) – The wrapped key

Note: Data will be converted to hexadecimal by calling:

```
to_hex(from_bytestring(data))
```

If you need to pass in raw hex data, call:

```
to_bytestring(from_hex(hex-data))
```

References:

- *to_hex()*
- *from_hex()*
- *to_bytestring()*
- *from_bytestring()*

- **key_template** (*dict*) – The python template representing the new key's template
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns (Retcode, unwrapped key handle)

Return type tuple

pycryptoki.encryption.**c_unwrap_key_ex**(*h_session*, *h_unwrapping_key*, *wrapped_key*, *key_template*, *mechanism*)

Executes *c_unwrap_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.3.4 Multipart Helper

pycryptoki.encryption.**do_multipart_operation**(*h_session*, *c_update_function*, *c_finalize_function*, *input_data_list*, *output_buffer=None*)

Some code which will do a multipart encrypt or decrypt since they are the same with just different functions called

Parameters

- **h_session** (*int*) – Session handle
- **c_update_function** – C_<NAME>Update function to call to update each operation.
- **c_finalize_function** – Function to call at end of multipart operation.
- **input_data_list** – List of data to call update function on.

Note: Data will be converted to hexadecimal by calling:

```
to_hex(from_bytestring(data))
```

If you need to pass in raw hex data, call:

```
to_bytestring(from_hex(hex-data))
```

References:

- *to_hex()*
- *from_hex()*
- *to_bytestring()*
- *from_bytestring()*

- **output_buffer** (*list*) – List of integers that specify a size of output buffers to use for multi-part operations. By default will query with NULL pointer buffer to get required size of buffer

1.4.4 Sign/Verify operations

Contents

1.4.4.1 Sign

pycryptoki.sign_verify.**c_sign**(*h_session*, *h_key*, *data_to_sign*, *mechanism*, *output_buffer=None*)

Signs the given data with given key and mechanism.

Note: If data is a list or tuple of strings, multi-part operations will be used.

Parameters

- **h_session** (*int*) – Session handle
- **data_to_sign** – The data to sign, either a string or a list of strings. If this is a list a multipart operation will be used (using C_...Update and C_...Final)

  ex:

  – "This is a proper argument of some data to use in the function"

  – ["This is another format of data this", "function will accept.", "It will operate on these strings in parts"]
- **h_key** (*int*) – The signing key
- **mechanism** – See the `parse_mechanism()` function for possible values.
- **output_buffer** (*list|int*) – Integer or list of integers that specify a size of output buffer to use for an operation. By default will query with NULL pointer buffer to get required size of buffer.

Returns (retcode, python string of signed data)

Return type tuple

`pycryptoki.sign_verify.`**`c_sign_ex`**(*h_session*, *h_key*, *data_to_sign*, *mechanism*, *output_buffer=None*)

Executes *c_sign()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.4.2 Verify

`pycryptoki.sign_verify.`**`c_verify`**(*h_session*, *h_key*, *data_to_verify*, *signature*, *mechanism*)

Verifies data with the given signature, key and mechanism.

Note: If data is a list or tuple of strings, multi-part operations will be used.

Parameters

- **h_session** (*int*) – Session handle

- **data_to_verify** – The data to sign, either a string or a list of strings. If this is a list a multipart operation will be used (using C_...Update and C_...Final)

  ex:

  – "This is a proper argument of some data to use in the function"

  – ["This is another format of data this", "function will accept.", "It will operate on these strings in parts"]

- **signature** (*bytes*) – Signature with which to verify the data.
- **h_key** (*int*) – The verifying key
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns retcode of verify operation

pycryptoki.sign_verify.**c_verify_ex**(*h_session*, *h_key*, *data_to_verify*, *signature*, *mechanism*)
Executes *c_verify()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.sign_verify.**do_multipart_verify**(*h_session*, *input_data_list*, *signature*)
Do a multipart verify operation

Parameters

- **h_session** (*int*) – Session handle
- **input_data_list** – list of data to verify with
- **signature** – signature to verify

Returns The result code

1.4.5 Attributes and Conversions

Contents

This module contains a wrapper around the key attributes and the template struct generation to make it possible to create templates in python and easily convert them into templates in C.

pycryptoki.attributes.**KEY_TRANSFORMS CK_ATTRIBUTE Types mapped to Python->C transformation**

pycryptoki.attributes.**ret_type**(*c_type*)
Decorator to set a returned C Type so we can determine what type to use for an AutoCArray

Parameters **c_type** – Default return-type of the transform function.

pycryptoki.attributes.**to_long**(*val*, *reverse=False*)
Convert a integer/long value to a pValue, ulValueLen tuple

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to ctypes.c_ulong, ctypes.c_ulong

size of long value)

pycryptoki.attributes.**to_bool**(*val*, *reverse=False*)
Convert a boolean-ish value to a pValue, ulValueLen tuple.

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to *pycryptoki.cryptoki.CK_BBOOL*,

ctypes.c_ulong size of bool value)

pycryptoki.attributes.**to_char_array**(*val*, *reverse=False*)
Convert the given string or list of string values into a char array.

This is slightly different than to_byte_array, which has different assumptions as to the format of the input.

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to *pycryptoki.cryptoki.CK_CHAR* array,

ctypes.c_ulong size of array)

pycryptoki.attributes.**to_ck_date**(*val*, *reverse=False*)
Transform a date string, date dictionary, or date object into a PKCS11 readable form (YYYYMMDD)

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to *pycryptoki.cryptoki.CK_CHAR* array,

ctypes.c_ulong size of array)

pycryptoki.attributes.**to_pka_key_status**(*val*, *reverse=False*)
Transform a Per Key Authorization Key Status object into a PKCS11 readable byte string

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to pycryptoki.cryptoki.CK_KEY_STATUS object,

ctypes.c_ulong size of array)

pycryptoki.attributes.**to_byte_array**(*val*, *reverse=False*)
Converts an arbitrarily sized integer, list, or byte array into a byte array.

It'll zero-pad the bit length so it's a multiple of 8, then convert the int to binary, split the binary string into sections of 8, then place each section into a slot in a ctypes.c_ubyte array (converting to small int).

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to *pycryptoki.cryptoki.CK_BYTE* array,

ctypes.c_ulong size of array)

pycryptoki.attributes.**to_sub_attributes**(*val*, *reverse=False*)
Convert to another Attributes class & return the struct.

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to *pycryptoki.cryptoki.CK_ATTRIBUTE* array,

ctypes.c_ulong size of array)

class pycryptoki.attributes.**Attributes**(**args*, ***kwargs*)
Python container for handling PKCS11 Attributes.

Provides *get_c_struct()*, that would returns a list of C Structs, each with the following structure:

```
class CK_ATTRIBUTE(Structure):
    '''
    Defines type, value and length of an attribute:

    c_ulong type;
    c_void_p pValue;
    c_ulong ulValueLen;
    '''
    pass
```

This list of structs can be used with *C_GetAttributeValue()* to get the length of the value that will be placed in pValue (will be set to ulValueLen), or if you already know the length required you can 'blank fill' pValue for direct use.

You can also provide new transformations in the form of a dictionary that will be preferred to the *KEY_TRANSFORMS* dictionary. This is passed in only as a keyword argument:

```
transform = {1L: lambda x: return x**2}`
attrs = Attributes({...}, new_transforms=transform)
# attrs.get_c_struct will use the lambda expression in the transform dictionary
# for key 1L
```

get_c_struct()
Build an array of *CK_ATTRIBUTE* Structs & return it.

Returns *CK_ATTRIBUTE* array

static from_c_struct(*c_struct*)
Build out a dictionary from a c_struct.

Parameters **c_struct** – Pointer to an array of *CK_ATTRIBUTE* structs

Returns dict

pycryptoki.attributes.**c_struct_to_python**(*c_struct*)
Converts a C struct to a python dictionary.

Parameters **c_struct** – The c struct to convert into a dictionary in python

Returns Returns a python dictionary which represents the C struct passed in

pycryptoki.attributes.**convert_c_ubyte_array_to_string**(*byte_array*)
Converts a ctypes unsigned byte array into a string.

Parameters **byte_array** –

1.4.5.1 Conversions

Provide low-level conversions between common data types.

The `from_xyz` functions should all return an iterator over a list of integers, representing the individual bytes in the passed-in value.

The `to_xyz` functions take in an iterable of integers and convert it to the specified type.

Example 1

Listing 1: Convert a raw bytestring to hex

```
raw_bytes = from_bytestring(b"Some test data")
assert raw_bytes = [83, 111, 109, 101, 32, 116, 101, 115, 116, 32, 100, 97,␣
↪116, 97]

hex_data = to_hex(from_bytestring(b"Some test data"))
assert hex_data == b'536f6d6520746573742064617461'
```

Example 2

Listing 2: Convert hex data to a raw bytestring

```
bytestring_data = to_bytestring(from_hex(b'536f6d6520746573742064617461'))
assert bytestring_data == b"Some test data"

raw_bytes = list(from_hex(b'536f6d6520746573742064617461'))
assert raw_bytes == [83, 111, 109, 101, 32, 116, 101, 115, 116, 32, 100, 97,␣
↪116, 97]
```

pycryptoki.conversions.**from_bytestring**(*ascii_*)
Convert an iterable of strings into an iterable of integers.

Note: For bytestrings on python3, this does effectively nothing, since iterating over a bytestring in python 3 will return integers.

Parameters **ascii** – String to convert

Returns iterator

pycryptoki.conversions.**to_bytestring**(*ascii_*)
Convert an iterable of integers into a bytestring.

Parameters **ascii** (*iterable*) – Iterable of integers

Returns bytestring

pycryptoki.conversions.**from_bin**(*bin_*)
Convert a string-representation of binary into a list of integers.

Parameters **bin** (*str*) – String representation of binary data (ex: "10110111")

Returns iterator over integers

pycryptoki.conversions.**to_bin**(*ascii_*)
Convert an iterable of integers to a binary representation.

Parameters **ascii** (*iterable*) – iterable of integers

Returns bytestring of the binary values

pycryptoki.conversions.**from_hex**(*hex_*)
Convert a hexademical string to an iterable of integers.

Parameters **hex** (*str*) – Hex string

Returns Iterator

pycryptoki.conversions.**to_hex**(*ints*)
Convert an iterable of integers to a hexadecimal string.

Parameters **ints** (*iterable*) – Iterable of integers

Returns bytestring representing the hex data.

1.4.6 Mechanisms

Contents

Conversions for pure-python dictionaries to C struct mechanisms.

To implement a new Mechanism:

1. Create a new mechanism class, deriving from *Mechanism*
2. Set REQUIRED_PARAMS as a class variable. REQUIRED_PARAMS should be a list of strings, defining required parameter keys.

```
class IvMechanism(Mechanism):
    REQUIRED_PARAMS = ['iv']
```

3. Override `to_c_mech()` on the new mechanism class. This function can access `self.params` to get passed-in parameters, and should create the C parameter struct required by the mechanism. This should also return `self.mech` (which is a `CK_MECHANISM` struct).

Listing 3: Simple Example

```
class IvMechanism(Mechanism):
    REQUIRED_PARAMS = ['iv']

    def to_c_mech(self):
        super(IvMechanism, self).to_c_mech()
        if len(self.params['iv']) == 0:
            LOG.debug("Setting IV to NULL (using internal)")
            iv_ba = None
            iv_len = 0
        else:
            iv_ba, iv_len = to_byte_array(self.params['iv'])
        self.mech.pParameter = iv_ba
        self.mech.usParameterLen = iv_len
        return self.mech
```

Listing 4: Example with a PARAMS struct

```
class AESXTSMechanism(Mechanism):
    REQUIRED_PARAMS = ['cb', 'hTweakKey']

    def to_c_mech(self):
        super(AESXTSMechanism, self).to_c_mech()
        xts_params = CK_AES_XTS_PARAMS()
        xts_params.cb = (CK_BYTE * 16)(*self.params['cb'])
        xts_params.hTweakKey = CK_ULONG(self.params['hTweakKey'])
        self.mech.pParameter = cast(pointer(xts_params), c_void_p)
        self.mech.usParameterLen = CK_ULONG(sizeof(xts_params))
        return self.mech
```

1.4.6.1 Helpers

Mechanism base class, as well as helper functions for parsing Mechanism arguments to pycryptoki functions.

class pycryptoki.mechanism.helpers.**Mechanism**(*mech_type='UNKNOWN'*, *params=None*)
Bases: object

Base class for pycryptoki mechanisms. Performs checks for missing parameters w/ created mechs, and creates the base Mechanism Struct for conversion to ctypes.

REQUIRED_PARAMS = []

to_c_mech()
Create the Mechanism structure & set the mech type to the passed-in flavor.

Returns *CK_MECHANISM*

exception pycryptoki.mechanism.helpers.**MechanismException**
Bases: Exception

Exception raised for mechanism errors. Ex: required parameters are missing

pycryptoki.mechanism.helpers.**get_c_struct_from_mechanism**(*python_dictionary*, *params_type_string*)

Gets a c struct from a python dictionary representing that struct

Parameters

- **python_dictionary** – The python dictionary representing the C struct, see CK_AES_CBC_PAD_EXTRACT_PARAMS for an example
- **params_type_string** – A string representing the parameter struct. ex. for *CK_AES_CBC_PAD_EXTRACT_PARAMS* use the string CK_AES_CBC_PAD_EXTRACT_PARAMS

Returns A C struct

pycryptoki.mechanism.helpers.**get_python_dict_from_c_mechanism**(*c_mechanism*, *params_type_string*)

Gets a python dictionary from a c mechanism's struct for serialization and easier test case writing

Parameters

- **c_mechanism** – The c mechanism to convert to a python dictionary
- **params_type_string** – A string representing the parameter struct. ex. for *CK_AES_CBC_PAD_EXTRACT_PARAMS* use the string CK_AES_CBC_PAD_EXTRACT_PARAMS

Returns A python dictionary representing the c struct

pycryptoki.mechanism.helpers.**parse_mechanism**(*mechanism_param*)

Designed for use with any function call that takes in a mechanism, this will handle a mechanism parameter that is one of the following:

1. CKM_ integer constant – will create a *CK_MECHANISM* with only mech_type set.

```
parse_mechanism(CKM_RSA_PKCS)
# Results in:
mech = CK_MECHANISM()
mech.mechanism = CK_MECHANISM_TYPE(CKM_RSA_PKCS)
mech.pParameter = None
mech.usParameterLen = 0
```

2. Dictionary with mech_type as a mandatory key, and params as an optional key. This will be passed into the *Mechanism* class for conversion to a *CK_MECHANISM*.

```
parse_mechanism({'mech_type': CKM_AES_CBC,
                 'params': {'iv': list(range(8))}})
# Results in:
mech = CK_MECHANISM()
mech.mechanism = CK_MECHANISM_TYPE(CKM_AES_CBC)
iv_ba, iv_len = to_byte_array(list(range(8)))
mech.pParameter = iv_ba
mech.usParameterLen = iv_len
```

3. *CK_MECHANISM* struct – passed directly into the raw C Call.
4. Mechanism class – will call to_c_mech() on the class, and use the results.

Warning: If you're using this with rpyc, you need to make sure the call *to_c_mech* occurs on the *server* (the machine with the HSM)! If you pass in a *Mechanism* class that was created on the client, the resulting call into *to_c_mech()* will *also* be on the client side!

Note: You can look at `REQUIRED_PARAMS` on each mechanism class to see what parameters are required.

Parameters **mechanism_param** – Parameter to convert to a C Mechanism.

Returns *CK_MECHANISM* struct.

1.4.6.2 AES Mechanisms

AES-specific mechanism implementations.

class pycryptoki.mechanism.aes.**AESCBCEncryptDataMechanism**(*mech_type='UNKNOWN'*, *params=None*)

Bases: *pycryptoki.mechanism.helpers.Mechanism*

AES CBC mechanism for deriving keys from encrypted data.

REQUIRED_PARAMS = ['iv', 'data']

to_c_mech()

Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.aes.**AESCTRMechanism**(*mech_type='UNKNOWN'*, *params=None*)

Bases: *pycryptoki.mechanism.helpers.Mechanism*

AES CTR Mechanism param conversion.

REQUIRED_PARAMS = ['cb', 'ulCounterBits']

to_c_mech()

Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.aes.**AESECBEncryptDataMechanism**(*mech_type='UNKNOWN'*, *params=None*)

Bases: *pycryptoki.mechanism.helpers.Mechanism*

AES mechanism for deriving keys from encrypted data.

REQUIRED_PARAMS = ['data']

to_c_mech()

Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.aes.**AESGCMMechanism**(*mech_type='UNKNOWN'*, *params=None*)

Bases: *pycryptoki.mechanism.helpers.Mechanism*

Creates the AES-GCM specific param structure & converts python types to C types.

REQUIRED_PARAMS = ['iv', 'AAD', 'ulTagBits']

to_c_mech()
Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.aes.**AESXTSMechanism**(*mech_type='UNKNOWN'*, *params=None*)
Bases: *pycryptoki.mechanism.helpers.Mechanism*

Creates the AES-XTS specific param structure & converts python types to C types.

REQUIRED_PARAMS = ['cb', 'hTweakKey']

to_c_mech()
Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.aes.**Iv16Mechanism**(*mech_type='UNKNOWN'*, *params=None*)
Bases: *pycryptoki.mechanism.helpers.Mechanism*

Mech class for flavors that require an IV set in the mechanism. Will default to *[1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8]* if no IV is passed in

to_c_mech()
Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.aes.**IvMechanism**(*mech_type='UNKNOWN'*, *params=None*)
Bases: *pycryptoki.mechanism.helpers.Mechanism*

Mech class for flavors that require an IV set in the mechanism. Will default to *[0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38]* if no IV is passed in

to_c_mech()
Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

1.4.6.3 Generic Mechanisms

Generic Mechanisms conversions.

class pycryptoki.mechanism.generic.**AutoMech**(*mech_type='UNKNOWN'*, *params=None*)
Bases: *pycryptoki.mechanism.helpers.Mechanism*

An attempt to examine underlying C Struct and fill in the appropriate fields, making some assumptions about the data. This works best with parameter structs that only have CK_ULONGs within them (though there is a best-effort attempt to handle arrays).

> **Warning:** Do not use this if the mechanism is already defined!

to_c_mech()
Attempt to handle generic mechanisms by introspection of the structure.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.generic.**ConcatenationDeriveMechanism**(*mech_type='UNKNOWN'*, *params=None*)
Bases: *pycryptoki.mechanism.helpers.Mechanism*

Mechanism class for key derivations. This will take in a second key handle in the parameters, and use it in the resulting Structure.

> **Warning:** This mechanism is disabled in later versions of PCKS11.

REQUIRED_PARAMS = ['h_second_key']

to_c_mech()
Add in a pointer to the second key in the resulting mech structure.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.generic.**NullMech**(*mech_type='UNKNOWN'*, *params=None*)
Bases: *pycryptoki.mechanism.helpers.Mechanism*

Class that creates a mechanism from a flavor with null parameters. Used mostly for signing mechanisms that really don't need anything else.

to_c_mech()
Simply set the pParameter to null pointer.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.generic.**StringDataDerivationMechanism**(*mech_type='UNKNOWN'*, *params=None*)
Bases: *pycryptoki.mechanism.helpers.Mechanism*

Mechanism class for key derivation using passed in string data.

REQUIRED_PARAMS = ['data']

to_c_mech()
Convert data to bytearray, then use in the resulting mech structure.

Returns *CK_MECHANISM*

1.4.6.4 RC Mechanisms

RC-related Mechanism implementations

class pycryptoki.mechanism.rc.**RC2CBCMechanism**(*mech_type='UNKNOWN'*, *params=None*)
Bases: *pycryptoki.mechanism.helpers.Mechanism*

Creates required RC2CBC Param structure & converts python data to C data.

REQUIRED_PARAMS = ['usEffectiveBits', 'iv']

to_c_mech()
Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.rc.**RC2Mechanism**(*mech_type='UNKNOWN'*, *params=None*)
Bases: *pycryptoki.mechanism.helpers.Mechanism*

Sets the mechanism parameter to the usEffectiveBits

REQUIRED_PARAMS = ['usEffectiveBits']

to_c_mech()
Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.rc.**RC5CBCMechanism**(*mech_type='UNKNOWN'*, *params=None*)

Bases: *pycryptoki.mechanism.helpers.Mechanism*

Creates required RC5CBC Param structure & converts python data to C data.

REQUIRED_PARAMS = ['ulWordsize', 'ulRounds', 'iv']

to_c_mech()

Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.rc.**RC5Mechanism**(*mech_type='UNKNOWN'*, *params=None*)

Bases: *pycryptoki.mechanism.helpers.Mechanism*

Creates required RC5 Param structure & converts python data to C data.

REQUIRED_PARAMS = ['ulWordsize', 'ulRounds']

to_c_mech()

Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

1.4.6.5 RSA Mechanisms

RSA-related Mechanism implementations.

class pycryptoki.mechanism.rsa.**RSAPKCSOAEPMechanism**(*mech_type='UNKNOWN'*, *params=None*)

Bases: *pycryptoki.mechanism.helpers.Mechanism*

Create the required RSA_PKCS_OAEP param structure & convert python data to C data.

REQUIRED_PARAMS = ['hashAlg', 'mgf']

to_c_mech()

Convert extra parameters to ctypes, then build out the mechanism.

Returns *CK_MECHANISM*

class pycryptoki.mechanism.rsa.**RSAPKCSPSSMechanism**(*mech_type='UNKNOWN'*, *params=None*)

Bases: *pycryptoki.mechanism.helpers.Mechanism*

Create the required RSA_PKCS_PSS param structure & convert python data to C data.

REQUIRED_PARAMS = ['hashAlg', 'mgf']

to_c_mech()

Uses default salt length of 8. Can be overridden w/ a parameter though.

Returns *CK_MECHANISM*

1.4.7 Miscellaneous

Contents

1.4.7.1 RNG, Digest, Creating Objects

PKCS11 Interface to the following functions:

- c_generate_random
- c_seed_random
- c_digest
- c_digestkey
- c_create_object
- c_set_ped_id (**CA_** function)
- c_get_ped_id (**CA_** function)

pycryptoki.misc.**c_generate_random**(*h_session*, *length*)
Generates a sequence of random numbers

Parameters

- **h_session** (*int*) – Session handle
- **length** (*int*) – The length in bytes of the random number sequence

Returns (retcode, A string of random data)

Return type tuple

pycryptoki.misc.**c_generate_random_ex**(*h_session*, *length*)
Executes *c_generate_random()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.misc.**c_seed_random**(*h_session*, *seed*)
Seeds the random number generator

Parameters

- **h_session** (*int*) – Session handle
- **seed** (*bytes*) – A python string of some seed

Returns retcode

Return type int

pycryptoki.misc.**c_seed_random_ex**(*h_session*, *seed*)
Executes *c_seed_random()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.misc.**c_digest**(*h_session*, *data_to_digest*, *digest_flavor*, *mechanism=None*, *output_buffer=None*)
Digests some data

Parameters

- **h_session** (*int*) – Session handle
- **data_to_digest** (*bytes*) – The data to digest, either a string or a list of strings. If this is a list a multipart operation will be used
- **digest_flavor** (*int*) – The flavour of the mechanism to digest (MD2, SHA-1, HAS-160, SHA224, SHA256, SHA384, SHA512)
- **mechanism** – See the `parse_mechanism()` function for possible values. If None will use digest flavor.
- **output_buffer** (*list | int*) – Integer or list of integers that specify a size of output buffer to use for an operation. By default will query with NULL pointer buffer to get required size of buffer.

Returns (retcode, a python string of the digested data)

Return type tuple

pycryptoki.misc.**c_digest_ex**(*h_session*, *data_to_digest*, *digest_flavor*, *mechanism=None*, *output_buffer=None*)
Executes *c_digest()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.misc.**c_digestkey**(*h_session*, *h_key*, *digest_flavor*, *mechanism=None*)
Digest a key

Parameters

- **h_session** (*int*) – Session handle
- **h_key** (*int*) – Key to digest
- **digest_flavor** (*int*) – Digest flavor
- **mechanism** – See the `parse_mechanism()` function for possible values. If None will use digest flavor.

pycryptoki.misc.**c_digestkey_ex**(*h_session*, *h_key*, *digest_flavor*, *mechanism=None*)
Executes *c_digestkey()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.misc.**c_create_object**(*h_session*, *template*)
Creates an object based on a given python template

Parameters

- **h_session** (*int*) – Session handle
- **template** (*dict*) – The python template which the object will be based on

Returns (retcode, the handle of the object)

Return type tuple

pycryptoki.misc.**c_create_object_ex**(*h_session*, *template*)
Executes *c_create_object()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.misc.**c_set_ped_id**(*slot*, *id*)
Set the PED ID for the given slot.

Parameters

- **slot** – slot number
- **id** – PED ID to use

Returns The result code

pycryptoki.misc.**c_set_ped_id_ex**(*slot*, *id*)
Executes *c_set_ped_id()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.misc.**c_get_ped_id**(*slot*)
Get the PED ID for the given slot.

Parameters **slot** – slot number

Returns The result code and ID

pycryptoki.misc.**c_get_ped_id_ex**(*slot*)
Executes *c_get_ped_id()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.7.2 Find Objects, Attribute Setting/Getting

Functions for dealing with object attributes

pycryptoki.object_attr_lookup.**c_find_objects**(*h_session*, *template*, *num_entries*)
Calls c_find_objects and c_find_objects_init to get a python dictionary of the objects found.

Parameters

- **h_session** (*int*) – Session handle
- **template** – A python dictionary of the object template to look for
- **num_entries** – The max number of entries to return

Returns Returns a list of handles of objects found

pycryptoki.object_attr_lookup.**c_find_objects_ex**(*h_session*, *template*, *num_entries*)
Executes *c_find_objects()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.object_attr_lookup.**c_get_attribute_value**(*h_session*, *h_object*, *template*)
Calls C_GetAttrributeValue to get an attribute value based on a python template

Parameters

- **h_session** (*int*) – Session handle
- **h_object** – The handle of the object to get attributes for
- **template** – A python dictionary representing the template of the attributes to be retrieved

Returns A python dictionary representing the attributes returned from the HSM/library

pycryptoki.object_attr_lookup.**c_get_attribute_value_ex**(*h_session*, *h_object*, *template*)
Executes *c_get_attribute_value()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.object_attr_lookup.**c_set_attribute_value**(*h_session*, *h_object*, *template*)
Calls C_SetAttributeValue to set an attribute value based on a python template

Parameters

- **h_session** (*int*) – Session handle
- **h_object** – The handle of the object to get attributes for
- **template** – A python dictionary representing the template of the attributes to be written

Returns A python dictionary representing the attributes returned from the HSM/library

pycryptoki.object_attr_lookup.**c_set_attribute_value_ex**(*h_session*, *h_object*, *template*)
Executes *c_set_attribute_value()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.7.3 HSM Management

Methods responsible for pycryptoki 'hsm management' set of commands.

pycryptoki.hsm_management.**c_performselftest**(*slot*, *test_type*, *input_data*, *input_data_len*)
Test: Performs a self test for specified test type on a given slot.

Parameters

- **slot** – slot number
- **test_type** – type of test CK_ULONG
- **input_data** – pointer to input data CK_BYTE_PTR
- **input_data_len** – input data length CK_ULONG

Returns

the result code

[CK_SLOT_ID, CK_ULONG, CK_BYTE_PTR, CK_ULONG, CK_BYTE_PTR, CK_ULONG_PTR]

pycryptoki.hsm_management.**c_performselftest_ex**(*slot*, *test_type*, *input_data*, *input_data_len*)

Executes *c_performselftest()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.hsm_management.**ca_settokencertificatesignature**(*h_session*, *access_level*, *customer_id*, *pub_template*, *signature*, *signature_len*)

Completes the installation of a certificate on a token. The caller must supply a public key and a signature for token certificate. The public key is provided through the template; it must contain a key type, a modulus and a public exponent.

Parameters

- **h_session** (*int*) – Session handle
- **access_level** – the access level
- **customer_id** – the customer ID
- **pub_template** – the public template
- **signature** – the signature
- **signature_len** – the length in bytes of the signature

Returns the result code

pycryptoki.hsm_management.**ca_settokencertificatesignature_ex**(*h_session*, *access_level*, *customer_id*, *pub_template*, *signature*, *signature_len*)

Executes *ca_settokencertificatesignature()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.hsm_management.**ca_hainit**(*h_session*, *h_key*)
Creates a login key pair on the primary token.

Parameters

- **h_session** (*int*) – Session handle
- **h_key** – the login private key

Returns the result code

pycryptoki.hsm_management.**ca_hainit_ex**(*h_session*, *h_key*)
Executes *ca_hainit()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.hsm_management.**ca_createloginchallenge**(*h_session*, *user_type*, *challenge*)
Creates a login challenge for the given user.

Parameters

- **h_session** (*int*) – Session handle
- **user_type** – user type
- **challenge** – challenge

Returns the result code

pycryptoki.hsm_management.**ca_createloginchallenge_ex**(*h_session*, *user_type*, *challenge*)
Executes *ca_createloginchallenge()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.hsm_management.**ca_initializeremotepedvector**(*h_session*)
Initializes a remote PED vector

Parameters **h_session** (*int*) – Session handle

Returns the result code

pycryptoki.hsm_management.**ca_initializeremotepedvector_ex**(*h_session*)
Executes *ca_initializeremotepedvector()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.hsm_management.**ca_deleteremotepedvector**(*h_session*)
Deletes a remote PED vector

Parameters **h_session** (*int*) – Session handle

Returns the result code

pycryptoki.hsm_management.**ca_deleteremotepedvector_ex**(*h_session*)
Executes *ca_deleteremotepedvector()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.hsm_management.**ca_mtkrestore**(*slot*)
Restore the MTK

Parameters **slot** – slot number

Returns the result code

pycryptoki.hsm_management.**ca_mtkrestore_ex**(*slot*)
Executes *ca_mtkrestore()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.hsm_management.**ca_mtkresplit**(*slot*)
Resplit the MTK

Parameters **slot** – slot number

Returns the result code

pycryptoki.hsm_management.**ca_mtkresplit_ex**(*slot*)
Executes *ca_mtkresplit()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.hsm_management.**ca_mtkzeroize**(*slot*)
Zeroize the MTK

Parameters **slot** – slot number

Returns the result code

pycryptoki.hsm_management.**ca_mtkzeroize_ex**(*slot*)
Executes *ca_mtkzeroize()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.hsm_management.**ca_set_hsm_policy**(*h_session*, *policy_id*, *policy_val*)
Sets the HSM policies by calling CA_SetHSMPolicy

Parameters

- **h_session** (*int*) – Session handle
- **policy_id** – The ID of the policy being set
- **policy_val** – The value of the policy being set

Returns The result code

pycryptoki.hsm_management.**ca_set_hsm_policy_ex**(*h_session*, *policy_id*, *policy_val*)
Executes *ca_set_hsm_policy()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.hsm_management.**ca_set_hsm_policies**(*h_session*, *policies*)
Set multiple HSM policies.

Parameters

- **h_session** (*int*) – Session handle
- **policies** – dict of policy ID ints and value ints

Returns result code

pycryptoki.hsm_management.**ca_set_hsm_policies_ex**(*h_session*, *policies*)
Executes *ca_set_hsm_policies()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.hsm_management.**ca_set_destructive_hsm_policy**(*h_session*, *policy_id*, *policy_val*)

Sets the destructive HSM policies by calling CA_SetDestructiveHSMPolicy

Parameters

- **h_session** (*int*) – Session handle
- **policy_id** – The ID of the policy being set
- **policy_val** – The value of the policy being set

Returns The result code

pycryptoki.hsm_management.**ca_set_destructive_hsm_policy_ex**(*h_session*, *policy_id*, *policy_val*)

Executes *ca_set_destructive_hsm_policy()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.hsm_management.**ca_set_destructive_hsm_policies**(*h_session*, *policies*)

Set multiple HSM policies.

Parameters

- **h_session** (*int*) – Session handle
- **policies** – dict of policy ID ints and value ints

Returns result code

pycryptoki.hsm_management.**ca_set_destructive_hsm_policies_ex**(*h_session*, *policies*)

Executes *ca_set_destructive_hsm_policies()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.hsm_management.**ca_get_hsm_capability_set**(*slot*)

Get the capabilities of the given slot.

Parameters **slot** (*int*) – Target slot number

Returns retcode, {id: val} dict of capabilities (None if command failed)

pycryptoki.hsm_management.**ca_get_hsm_capability_set_ex**(*slot*)

Executes *ca_get_hsm_capability_set()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.hsm_management.**ca_get_hsm_capability_setting**(*slot*, *capability_id*)

Get the value of a single capability

Parameters

- **slot** – slot ID of slot to query
- **capability_id** – capability ID

Returns result code, CK_ULONG representing capability active or not

pycryptoki.hsm_management.**ca_get_hsm_capability_setting_ex**(*slot*, *capability_id*)

Executes *ca_get_hsm_capability_setting()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.hsm_management.**ca_get_hsm_policy_set**(*slot*)
Get the policies of the given slot.

Parameters **slot** (*int*) – Target slot number

Returns retcode, {id: val} dict of policies (None if command failed)

pycryptoki.hsm_management.**ca_get_hsm_policy_set_ex**(*slot*)
Executes *ca_get_hsm_policy_set()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.hsm_management.**ca_get_hsm_policy_setting**(*slot*, *policy_id*)
Get the value of a single policy

Parameters

- **slot** – slot ID of slot to query
- **policy_id** – policy ID

Returns result code, CK_ULONG representing policy active or not

pycryptoki.hsm_management.**ca_get_hsm_policy_setting_ex**(*slot*, *policy_id*)
Executes *ca_get_hsm_policy_setting()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.7.4 Audit Functions

Methods responsible for managing a user's session and login/c_logout

pycryptoki.audit_handling.**ca_init_audit**(*slot*, *audit_pin*, *audit_label*)

Parameters

- **slot** –
- **audit_pin** –
- **audit_label** –

pycryptoki.audit_handling.**ca_init_audit_ex**(*slot*, *audit_pin*, *audit_label*)

Executes *ca_init_audit()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.audit_handling.**ca_time_sync**(*h_session*, *ultime*)

Parameters

- **h_session** (*int*) – Session handle
- **ultime** –

pycryptoki.audit_handling.**ca_time_sync_ex**(*h_session*, *ultime*)

Executes *ca_time_sync()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.audit_handling.**ca_get_time**(*h_session*)

Parameters **h_session** (*int*) – Session handle

pycryptoki.audit_handling.**ca_get_time_ex**(*h_session*)

Executes *ca_get_time()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.7.5 Backup Functions

Backup related commands

pycryptoki.backup.**ca_open_secure_token**(*h_session*, *storage_path*, *dev_ID*, *mode*)

Parameters

- **h_session** (*int*) – Session handle
- **storage_path** –
- **dev_ID** –
- **mode** –

pycryptoki.backup.**ca_open_secure_token_ex**(*h_session*, *storage_path*, *dev_ID*, *mode*)

Executes *ca_open_secure_token()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.backup.**ca_close_secure_token**(*h_session*, *h_ID*)

Parameters

- **h_session** (*int*) – Session handle
- **h_ID** –

pycryptoki.backup.**ca_close_secure_token_ex**(*h_session*, *h_ID*)

Executes *ca_close_secure_token()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.backup.**ca_extract**(*h_session*, *mechanism*)

Parameters

- **h_session** (*int*) – Session handle
- **mechanism** – See the `parse_mechanism()` function for possible values.

pycryptoki.backup.**ca_extract_ex**(*h_session*, *mechanism*)

Executes *ca_extract()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.backup.**ca_insert**(*h_session*, *mechanism*)

Parameters

- **h_session** (*int*) – Session handle
- **mechanism** – See the `parse_mechanism()` function for possible values.

pycryptoki.backup.**ca_insert_ex**(*h_session*, *mechanism*)

Executes *ca_insert()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.backup.**ca_sim_extract**(*h_session*, *key_handles*, *authform*, *auth_secrets=None*, *subset_size=0*, *delete_after_extract=False*)

Extract multiple keys to a wrapped blob. The returned blob can then be written into a file.

Parameters

- **h_session** (*int*) – Session handle
- **key_handles** (*list[int]*) – List of key handles to extract
- **authform** (*int*) – Type of authentication to use. See pycryptoki.backup.SIM_AUTH for details
- **auth_secrets** (*list(str)*) – Authorization secrets to use (Length will correspond to the N value in ckdemo)
- **subset_size** (*int*) – Subset size required for key use (Corresponds to the M value in ckdemo)
- **delete_after_extract** (*bool*) – If true, will destroy the original keys after they have been extracted.

Returns retcode, blob_data tuple.

pycryptoki.backup.**ca_sim_extract_ex**(*h_session*, *key_handles*, *authform*, *auth_secrets=None*, *subset_size=0*, *delete_after_extract=False*)

Executes *ca_sim_extract()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.backup.**ca_sim_insert**(*h_session*, *blob_data*, *authform*, *auth_secrets=None*)

Insert keys into the HSM from blob data that was wrapped off using SIM.

Parameters

- **h_session** (*int*) – Session handle
- **blob_data** (*str*) – Read in raw wrapped data. Typically read in from a file.
- **authform** (*int*) – Type of authentication to use. See pycryptoki.backup.SIM_AUTH for details
- **auth_secrets** (*list[str]*) – Authorization secrets to use (Length will correspond to the N value in ckdemo)

Returns retcode, keys tuple, where keys is a list of integers.

pycryptoki.backup.**ca_sim_insert_ex**(*h_session*, *blob_data*, *authform*, *auth_secrets=None*)

Executes *ca_sim_insert()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.backup.**ca_sim_multisign**(*h_session*, *blob_data*, *data_to_sign*, *mechanism*, *authform*, *auth_secrets=None*)

Sign data using keys that were extracted to a SIM blob.

Parameters

- **h_session** (*int*) – Session handle
- **blob_data** (*str*) – Read in raw wrapped key data. Typically read in from a file.
- **data_to_sign** – List of bytestring data to sign
- **mechanism** – Mechanism to use with the Sign operation
- **authform** (*int*) – Type of authentication to use. See pycryptoki.backup.SIM_AUTH for details
- **auth_secrets** (*list[str]*) – Authorization secrets to use (Length will correspond to the N value in ckdemo)

Returns retcode, signature list

pycryptoki.backup.**ca_sim_multisign_ex**(*h_session*, *blob_data*, *data_to_sign*, *mechanism*, *authform*, *auth_secrets=None*)

Executes *ca_sim_multisign()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.8 Pycryptoki Helpers

These are various helper modules and functions. They contain constant definitions, C parameter structs, configuration parsing, and default templates.

Contents

1.4.8.1 lookup_dicts

Module that contains lookup dictionaries for easy logging of error codes and other constants within pycryptoki.

pycryptoki.lookup_dicts.**ATTR_NAME_LOOKUP = {0: 'CKA_CLASS', 1: 'CKA_TOKEN', 2: 'CKA_PRIV**

pycryptoki.lookup_dicts.**ret_vals_dictionary = {0: 'CKR_OK', 1: 'CKR_CANCEL', 2: 'CKR_HOS**

1.4.8.2 default_templates

File containing a number of templates taken from CKDemo and manually converted into python format. See the attributes.py file for methods to convert them into the proper C format.

pycryptoki.default_templates.**CERTIFICATE_TEMPLATE = {0: 1, 1: True, 3: b'Created certif:**
The simple data object template taken from CKDemo when you select the Create Object option and choose data

pycryptoki.default_templates.**CKM_DH_PKCS_PARAMETER_GEN_TEMP = {1: True, 2: True, 3: b'SE**
The simple certificate object taken from CKDemo when you select the Create Object option and choose certificate

pycryptoki.default_templates.**CKM_SSL3_PRE_MASTER_KEY_GEN_TEMP = {1: True, 3: b'SSL3 Pre M**
Curve dictionary for ECDSA with oids as lists, taken from Components/tools/common/CommonData.cpp

pycryptoki.default_templates.**KEY_PAIR_GENERATOR_TEMPLATES = {0: ({1: True, 2: True, 368**
This list is not complete

pycryptoki.default_templates.**get_default_key_pair_template**(*mechanism*)
Gets the default template for the given key pair gen mechanism, returns a deep copy

Parameters **mechanism** –

pycryptoki.default_templates.**get_default_key_template**(*mechanism*)
Gets a default template for the given key gen mechanism, returns a deep copy

Parameters **mechanism** –

1.4.8.3 defaults

A file containing commonly used strings or other data similar to a config file

1.4.9 Extensions to the PKCS11 API

Thales-specific Extensions to the PKCS11 API.

Contents

1.4.9.1 Derive Key And Wrap

derive and wrap extended method

pycryptoki.ca_extensions.derive_wrap.**ca_derive_key_and_wrap**(*h_session*, *derive_mechanism*, *h_base_key*, *derive_template*, *wrapping_key*, *wrap_mechanism*, *output_buffer=2048*)
Derive a key from the base key and wrap it off the HSM using the wrapping key

Parameters

- **h_session** (*int*) – The session to use
- **h_base_key** (*int*) – The base key
- **derive_template** (*dict*) – A python template of attributes to set on derived key
- **derive_mechanism** – See the `parse_mechanism()` function for possible values.
- **wrapping_key** (*int*) – The wrapping key based on the encryption flavor
- **wrap_mechanism** – See the `parse_mechanism()` function for possible values.

- **output_buffer** – The size of the wrapped key, defaulted to a cert size

Returns (Retcode, python bytestring representing wrapped key)

Return type tuple

pycryptoki.ca_extensions.derive_wrap.**ca_derive_key_and_wrap_ex**(*h_session*, *derive_mechanism*, *h_base_key*, *derive_template*, *wrapping_key*, *wrap_mechanism*, *output_buffer=2048*)

Executes *ca_derive_key_and_wrap()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.9.2 HSM Info

Methods responsible for retrieving hsm info from the K7 card

pycryptoki.ca_extensions.hsm_info.**ca_retrieve_license_list**(*slot*)

Gets the license info for a given slot id

Parameters **slot_id** (*int*) – Slot index to get the license id's

Returns (A python list representing the license id's)

Return type list

pycryptoki.ca_extensions.hsm_info.**ca_retrieve_license_list_ex**(*slot*)

Executes *ca_retrieve_license_list()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.hsm_info.**ca_retrieve_allowed_containers**(*slot*)
Gets the maximum allowed container number for a given slot id

Parameters **slot_id** (*int*) – Slot index to get the maximum allowed container number

Returns (ret code, A unsigned integer representing the maximum allowed container number)

Return type unsigned integer

pycryptoki.ca_extensions.hsm_info.**ca_retrieve_allowed_containers_ex**(*slot*)
Executes *ca_retrieve_allowed_containers()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.hsm_info.**ca_retrieve_hsm_storage_info**(*slot*)
Gets the hsm storage info for a given slot id

Parameters **slot_id** (*int*) – Slot index to get the hsm storage info

Returns (ret code, hsm_storage_info dictionary)

Return type dictionary

pycryptoki.ca_extensions.hsm_info.**ca_retrieve_hsm_storage_info_ex**(*slot*)
Executes *ca_retrieve_hsm_storage_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.hsm_info.**ca_get_tsv**(*slot*)
Get the TSV(Module State Vector) for a given slot id

Parameters **slot_id** (*int*) – Slot index to get the TSV(Module State Vector)

Returns (ret code, TSV)

Return type tuple

pycryptoki.ca_extensions.hsm_info.**ca_get_tsv_ex**(*slot*)
Executes *ca_get_tsv()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.hsm_info.**ca_get_cv_firmware_version**(*slot_id*)
Cryptovisor specific ca extension function to get cv fw version

Parameters **slot_id** – slot id

Returns tuple of return code and cv fw version

pycryptoki.ca_extensions.hsm_info.**ca_get_cv_firmware_version_ex**(*slot_id*)
Executes *ca_get_cv_firmware_version()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.9.3 Object Commands

Module to work with objects, specifically dealing with ca_extension functions

pycryptoki.ca_extensions.object_handler.**ca_get_object_handle**(*slot*, *session*, *objectouid*)
Calls CA_GetObjectHandle to get the object handle from OUID

Parameters

- **slot** – partition slot number
- **session** – session id that was opened to run the function
- **objectouid** – OUID, a string of the hex value that maps to object handle

Returns a tuple containing the return code and the object handle mapping the given OUID

pycryptoki.ca_extensions.object_handler.**ca_get_object_handle_ex**(*slot*, *session*, *objectouid*)

Executes *ca_get_object_handle()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.object_handler.**ca_destroy_multiple_objects**(*h_session*, *objects*)

Delete multiple objects corresponding to given object handles

Parameters

- **h_session** (*int*) – Session handle
- **objects** (*list*) – The handles of the objects to delete

Returns Return code

pycryptoki.ca_extensions.object_handler.**ca_destroy_multiple_objects_ex**(*h_session*, *objects*)

Executes *ca_destroy_multiple_objects()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.9.4 Per Key Authorization

Module to work with PKA / Per key authorization

pycryptoki.ca_extensions.per_key_auth.**ca_set_authorization_data**(*h_session*, *h_object*, *old_auth_data*, *new_auth_data*)

User changes authorization data on key object (private, secret)

Parameters

- **h_session** – session handle
- **object** – key handle to update
- **old_auth_data** – byte list, e.g. [11, 12, 13, ..]
- **new_auth_data** – byte list, e.g. [11, 12, 13, ..]

Returns Ret code

pycryptoki.ca_extensions.per_key_auth.**ca_set_authorization_data_ex**(*h_session*, *h_object*, *old_auth_data*, *new_auth_data*)

Executes *ca_set_authorization_data()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.per_key_auth.**ca_reset_authorization_data**(*h_session*, *h_object*, *auth_data*)

CO resets auth data on unassigned key

Parameters

- **h_session** – session handle
- **object** – key handle to update
- **auth_data** – byte list, e.g. [11, 12, 13, ..]

Returns Ret code

pycryptoki.ca_extensions.per_key_auth.**ca_reset_authorization_data_ex**(*h_session*, *h_object*, *auth_data*)

Executes *ca_reset_authorization_data()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.ca_extensions.per_key_auth.**ca_increment_failed_auth_count**(*h_session*, *h_object*)

This function is called by HA group when auth failure happens on a key to sync up status. Here its defined mostly for testing purposes :param h_session: session handle :param object: key handle to update :return: Ret code

pycryptoki.ca_extensions.per_key_auth.**ca_increment_failed_auth_count_ex**(*h_session*, *h_object*)

Executes *ca_increment_failed_auth_count()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

pycryptoki.ca_extensions.per_key_auth.**ca_authorize_key**(*h_session*, *h_object*, *auth_data*)

User authorizes key within session or access for use

Parameters

- **h_session** – session handle
- **object** – key handle to authorize
- **auth_data** – authorization byte list, e.g. [11, 12, 13, ..]

Returns Ret code

pycryptoki.ca_extensions.per_key_auth.**ca_authorize_key_ex**(*h_session*, *h_object*, *auth_data*)

Executes *ca_authorize_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.per_key_auth.**ca_assign_key**(*h_session*, *h_object*)
Crypto Officer assigns a key

Parameters

- **h_session** – session handle
- **object** – key handle to assign

Returns Ret code

pycryptoki.ca_extensions.per_key_auth.**ca_assign_key_ex**(*h_session*, *h_object*)
Executes *ca_assign_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.9.5 Session Commands

Module to work with sessions, specifically dealing with ca_extension functions

pycryptoki.ca_extensions.session.**ca_get_session_info**(*session*)
ca extension function that returns session information

Parameters **session** – session handle

Returns tuple of return code and session info dict

pycryptoki.ca_extensions.session.**ca_get_session_info_ex**(*session*)
Executes *ca_get_session_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.session.**ca_get_application_id**()
Get the current process's AccessID.

Returns retcode, bytestring tuple.

pycryptoki.ca_extensions.session.**ca_get_application_id_ex**()
Executes *ca_get_application_id()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.session.**ca_open_application_id_v2**(*slot*, *appid*)
Open the given AccessID for the target slot.

Parameters

- **slot** – Slot #.
- **appid** – bytestring of length 16.

Returns Retcode.

pycryptoki.ca_extensions.session.**ca_open_application_id_v2_ex**(*slot*, *appid*)
Executes *ca_open_application_id_v2()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.session.**ca_close_application_id_v2**(*slot*, *appid*)
Close the AccessID associated with the given slot.

Parameters

- **slot** – Slot #.
- **appid** – bytestring of length 16.

Returns Retcode.

pycryptoki.ca_extensions.session.**ca_close_application_id_v2_ex**(*slot*, *appid*)
Executes *ca_close_application_id_v2()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.session.**ca_set_application_id_v2**(*appid*)
Set the Current process's AccessID.

Parameters **appid** – bytestring of length 16

Returns Retcode

pycryptoki.ca_extensions.session.**ca_set_application_id_v2_ex**(*appid*)
Executes *ca_set_application_id_v2()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.9.6 Utilization Metrics

Module to work with utilization metrics

pycryptoki.ca_extensions.utilization_metrics.**ca_read_utilization_metrics**(*session*)
HSM reads utilization data and saves as a snapshot

Parameters **session** – session id that was opened to run the function

Returns Ret code

pycryptoki.ca_extensions.utilization_metrics.**ca_read_utilization_metrics_ex**(*session*)
Executes *ca_read_utilization_metrics()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.utilization_metrics.**ca_read_and_reset_utilization_metrics**(*session*)
HSM reads current utilization data and saves as a snapshot; HSM resets metrics to zeroes

Parameters **session** – session id that was opened to run the function

Returns a dictionary with partition serial numbers as keys, value - dictionary of utilization metrics

pycryptoki.ca_extensions.utilization_metrics.**ca_read_and_reset_utilization_metrics_ex**(*session*
Executes *ca_read_and_reset_utilization_metrics()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

pycryptoki.ca_extensions.utilization_metrics.**ca_read_all_utilization_counters**(*h_session*)
Read Metrics from previously saved HSM snapshot Call either functions prior to create snapshot: ca_read_utilization_metrics ca_read_and_reset_utilization_metrics

Returns a dictionary, where keys are serial numbers

and values are dictionaries of bins and values, example: 'SIGN':0

pycryptoki.ca_extensions.utilization_metrics.**ca_read_all_utilization_counters_ex**(*h_session*)
Executes *ca_read_all_utilization_counters()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

1.4.10 Python/C Bindings

Definitions of PKCS11 types and Function bindings.

pycryptoki.cryptoki.**CA_ActivateMofN**(**args*)
Cryptoki DLL call to CA_ActivateMofN.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CA_MOFN_ACTIVATION
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CapabilityUpdate**(**args*)
Cryptoki DLL call to CA_CapabilityUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_CheckOperationState**(**args*)
Cryptoki DLL call to CA_CheckOperationState.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

- **arg3** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_ChoosePrimarySlot**(**args*)
Cryptoki DLL call to CA_ChoosePrimarySlot.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_ChooseSecondarySlot**(**args*)
Cryptoki DLL call to CA_ChooseSecondarySlot.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloneAllObjectsToSession**(**args*)
Cryptoki DLL call to CA_CloneAllObjectsToSession.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloneAsSource**(**args*)
Cryptoki DLL call to CA_CloneAsSource.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong
- **arg6** – c_ubyte
- **arg7** – LP_c_ubyte
- **arg8** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloneAsTarget**(**args*)
Cryptoki DLL call to CA_CloneAsTarget.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong
- **arg6** – c_ulong
- **arg7** – c_ulong

- **arg8** – c_ubyte
- **arg9** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloneAsTargetInit**(**args*)
Cryptoki DLL call to CA_CloneAsTargetInit.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong
- **arg6** – c_ubyte
- **arg7** – LP_c_ubyte
- **arg8** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloneModifyMofN**(**args*)
Cryptoki DLL call to CA_CloneModifyMofN.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_void_p

Returns c_ulong

pycryptoki.cryptoki.**CA_CloneMofN**(**args*)
Cryptoki DLL call to CA_CloneMofN.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_void_p

Returns c_ulong

pycryptoki.cryptoki.**CA_CloneObject**(**args*)
Cryptoki DLL call to CA_CloneObject.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloneObjectToAllSessions**(**args*)
Cryptoki DLL call to CA_CloneObjectToAllSessions.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_ClonePrivateKey**(**args*)
Cryptoki DLL call to CA_ClonePrivateKey.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloseAllSecondarySessions**(**args*)
Cryptoki DLL call to CA_CloseAllSecondarySessions.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloseApplicationID**(**args*)
Cryptoki DLL call to CA_CloseApplicationID.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloseApplicationIDForContainer**(**args*)
Cryptoki DLL call to CA_CloseApplicationIDForContainer.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloseApplicationIDV2**(**args*)
Cryptoki DLL call to CA_CloseApplicationIDV2.

Parameters

- **arg1** – c_ulong

- **arg2** – LP_CK_APPLICATION_ID

Returns c_ulong

pycryptoki.cryptoki.**CA_CloseSecondarySession**(**args*)
Cryptoki DLL call to CA_CloseSecondarySession.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CloseSecureToken**(**args*)
Cryptoki DLL call to CA_CloseSecureToken.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_ConfigureRemotePED**(**args*)
Cryptoki DLL call to CA_ConfigureRemotePED.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CreateContainer**(**args*)
Cryptoki DLL call to CA_CreateContainer.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – c_ulong
- **arg7** – c_ulong
- **arg8** – c_ulong
- **arg9** – c_ulong
- **arg10** – c_ulong
- **arg11** – c_ulong

- **arg12** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_CreateContainerLoginChallenge**(**args*)
Cryptoki DLL call to CA_CreateContainerLoginChallenge.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – LP_c_ulong
- **arg7** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_CreateContainerWithPolicy**(**args*)
Cryptoki DLL call to CA_CreateContainerWithPolicy.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – c_ulong
- **arg7** – c_ulong
- **arg8** – c_ulong
- **arg9** – c_ulong
- **arg10** – c_ulong
- **arg11** – c_ulong
- **arg12** – LP_c_ulong
- **arg13** – c_ulong
- **arg14** – c_ulong
- **arg15** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_CreateLoginChallenge**(**args*)
Cryptoki DLL call to CA_CreateLoginChallenge.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_Deactivate**(**args*)
Cryptoki DLL call to CA_Deactivate.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_DeactivateMofN**(**args*)
Cryptoki DLL call to CA_DeactivateMofN.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_DeleteContainer**(**args*)
Cryptoki DLL call to CA_DeleteContainer.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_DeleteContainerWithHandle**(**args*)
Cryptoki DLL call to CA_DeleteContainerWithHandle.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_DeleteRemotePEDVector**(**args*)
Cryptoki DLL call to CA_DeleteRemotePEDVector.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_DeriveKeyAndWrap**(**args*)
Cryptoki DLL call to CA_DeriveKeyAndWrap.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong
- **arg4** – LP_CK_ATTRIBUTE
- **arg5** – c_ulong
- **arg6** – LP_CK_MECHANISM

- **arg7** – c_ulong
- **arg8** – LP_c_ubyte
- **arg9** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_DestroyMultipleObjects**(**args*)
Cryptoki DLL call to CA_DestroyMultipleObjects.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_DisableUnauthTokenInsertion**(**args*)
Cryptoki DLL call to CA_DisableUnauthTokenInsertion.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_DismantleRemotePED**(**args*)
Cryptoki DLL call to CA_DismantleRemotePED.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_DuplicateMofN**(**args*)
Cryptoki DLL call to CA_DuplicateMofN.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_EnableUnauthTokenInsertion**(**args*)
Cryptoki DLL call to CA_EnableUnauthTokenInsertion.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_EncodeECChar2Params**(**args*)
Cryptoki DLL call to CA_EncodeECChar2Params.

Parameters

- **arg1** – LP_c_ubyte
- **arg2** – LP_c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – c_ulong
- **arg6** – c_ulong
- **arg7** – LP_c_ubyte
- **arg8** – c_ulong
- **arg9** – LP_c_ubyte
- **arg10** – c_ulong
- **arg11** – LP_c_ubyte
- **arg12** – c_ulong
- **arg13** – LP_c_ubyte
- **arg14** – c_ulong
- **arg15** – LP_c_ubyte
- **arg16** – c_ulong
- **arg17** – LP_c_ubyte
- **arg18** – c_ulong
- **arg19** – LP_c_ubyte
- **arg20** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_EncodeECParamsFromFile**(**args*)
Cryptoki DLL call to CA_EncodeECParamsFromFile.

Parameters

- **arg1** – LP_c_ubyte
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_EncodeECPrimeParams**(**args*)
Cryptoki DLL call to CA_EncodeECPrimeParams.

Parameters

- **arg1** – LP_c_ubyte
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – c_ulong

- **arg7** – LP_c_ubyte
- **arg8** – c_ulong
- **arg9** – LP_c_ubyte
- **arg10** – c_ulong
- **arg11** – LP_c_ubyte
- **arg12** – c_ulong
- **arg13** – LP_c_ubyte
- **arg14** – c_ulong
- **arg15** – LP_c_ubyte
- **arg16** – c_ulong
- **arg17** – LP_c_ubyte
- **arg18** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_Extract**(**args*)
Cryptoki DLL call to CA_Extract.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM

Returns c_ulong

pycryptoki.cryptoki.**CA_ExtractMaskedObject**(**args*)
Cryptoki DLL call to CA_ExtractMaskedObject.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_FactoryReset**(**args*)
Cryptoki DLL call to CA_FactoryReset.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_FindAdminSlotForSlot**(**args*)
Cryptoki DLL call to CA_FindAdminSlotForSlot.

Parameters

- **arg1** – c_ulong

- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_FirmwareRollback**(**args*)
Cryptoki DLL call to CA_FirmwareRollback.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_FirmwareUpdate**(**args*)
Cryptoki DLL call to CA_FirmwareUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong
- **arg6** – LP_c_ubyte
- **arg7** – c_ulong
- **arg8** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_GenerateCloneableMofN**(**args*)
Cryptoki DLL call to CA_GenerateCloneableMofN.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_CA_MOFN_GENERATION
- **arg4** – c_ulong
- **arg5** – c_ulong
- **arg6** – c_void_p

Returns c_ulong

pycryptoki.cryptoki.**CA_GenerateCloningKEV**(**args*)
Cryptoki DLL call to CA_GenerateCloningKEV.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GenerateMofN**(**args*)
Cryptoki DLL call to CA_GenerateMofN.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_CA_MOFN_GENERATION
- **arg4** – c_ulong
- **arg5** – c_ulong
- **arg6** – c_void_p

Returns c_ulong

pycryptoki.cryptoki.**CA_GenerateTokenKeys**(**args*)
Cryptoki DLL call to CA_GenerateTokenKeys.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_ATTRIBUTE
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_Get**(**args*)
Cryptoki DLL call to CA_Get.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetApplicationID**(**args*)
Cryptoki DLL call to CA_GetApplicationID.

Parameters **arg1** – LP_CK_APPLICATION_ID

Returns c_ulong

pycryptoki.cryptoki.**CA_GetCVFirmwareVersion**(**args*)
Cryptoki DLL call to CA_GetCVFirmwareVersion.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetClusterState**(**args*)
Cryptoki DLL call to CA_GetClusterState.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_CLUSTER_STATE

Returns c_ulong

pycryptoki.cryptoki.**CA_GetConfigurationElementDescription**(**args*)
Cryptoki DLL call to CA_GetConfigurationElementDescription.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ulong
- **arg7** – LP_c_ulong
- **arg8** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_GetContainerCapabilitySet**(**args*)
Cryptoki DLL call to CA_GetContainerCapabilitySet.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetContainerCapabilitySetting**(**args*)
Cryptoki DLL call to CA_GetContainerCapabilitySetting.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetContainerList**(**args*)
Cryptoki DLL call to CA_GetContainerList.

Parameters

- **arg1** – c_ulong

- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetContainerName**(**args*)
Cryptoki DLL call to CA_GetContainerName.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetContainerPolicySet**(**args*)
Cryptoki DLL call to CA_GetContainerPolicySet.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetContainerPolicySetting**(**args*)
Cryptoki DLL call to CA_GetContainerPolicySetting.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetContainerStatus**(**args*)
Cryptoki DLL call to CA_GetContainerStatus.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetContainerStorageInformation**(**args*)
Cryptoki DLL call to CA_GetContainerStorageInformation.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ulong
- **arg7** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetExtendedTPV**(**args*)
Cryptoki DLL call to CA_GetExtendedTPV.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetFPV**(**args*)
Cryptoki DLL call to CA_GetFPV.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetFunctionList**(**args*)
Cryptoki DLL call to CA_GetFunctionList.

Parameters **arg1** – LP_LP_CK_SFNT_CA_FUNCTION_LIST

Returns c_ulong

pycryptoki.cryptoki.**CA_GetHAState**(**args*)
Cryptoki DLL call to CA_GetHAState.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_HA_STATUS

Returns c_ulong

pycryptoki.cryptoki.**CA_GetHSMCapabilitySet**(**args*)
Cryptoki DLL call to CA_GetHSMCapabilitySet.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetHSMCapabilitySetting**(**args*)
Cryptoki DLL call to CA_GetHSMCapabilitySetting.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetHSMPolicySet**(**args*)
Cryptoki DLL call to CA_GetHSMPolicySet.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetHSMPolicySetting**(**args*)
Cryptoki DLL call to CA_GetHSMPolicySetting.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetHSMStats**(**args*)
Cryptoki DLL call to CA_GetHSMStats.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

- **arg4** – LP_HSM_STATS_PARAMS

Returns c_ulong

pycryptoki.cryptoki.**CA_GetHSMStorageInformation**(**args*)
Cryptoki DLL call to CA_GetHSMStorageInformation.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetModuleInfo**(**args*)
Cryptoki DLL call to CA_GetModuleInfo.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_CKCA_MODULE_INFO

Returns c_ulong

pycryptoki.cryptoki.**CA_GetModuleList**(**args*)
Cryptoki DLL call to CA_GetModuleList.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetMofNStatus**(**args*)
Cryptoki DLL call to CA_GetMofNStatus.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CA_M_OF_N_STATUS

Returns c_ulong

pycryptoki.cryptoki.**CA_GetNumberOfAllowedContainers**(**args*)
Cryptoki DLL call to CA_GetNumberOfAllowedContainers.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetObjectHandle**(**args*)
Cryptoki DLL call to CA_GetObjectHandle.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetObjectUID**(**args*)
Cryptoki DLL call to CA_GetObjectUID.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_GetPartitionPolicyTemplate**(**args*)
Cryptoki DLL call to CA_GetPartitionPolicyTemplate.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_GetPedId**(**args*)
Cryptoki DLL call to CA_GetPedId.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetPrimarySlot**(**args*)
Cryptoki DLL call to CA_GetPrimarySlot.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetRemotePEDVectorStatus**(**args*)
Cryptoki DLL call to CA_GetRemotePEDVectorStatus.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetRollbackFirmwareVersion**(**args*)
Cryptoki DLL call to CA_GetRollbackFirmwareVersion.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetSecondarySlot**(**args*)
Cryptoki DLL call to CA_GetSecondarySlot.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetSecureElementMeta**(**args*)
Cryptoki DLL call to CA_GetSecureElementMeta.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_CK_MECHANISM
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ubyte
- **arg7** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetServerInstanceBySlotID**(**args*)
Cryptoki DLL call to CA_GetServerInstanceBySlotID.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetSessionInfo**(**args*)
Cryptoki DLL call to CA_GetSessionInfo.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetSlotIdForContainer**(**args*)
Cryptoki DLL call to CA_GetSlotIdForContainer.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetSlotIdForPhysicalSlot**(**args*)
Cryptoki DLL call to CA_GetSlotIdForPhysicalSlot.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetSlotListFromServerInstance**(**args*)
Cryptoki DLL call to CA_GetSlotListFromServerInstance.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTPV**(**args*)
Cryptoki DLL call to CA_GetTPV.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTSV**(**args*)
Cryptoki DLL call to CA_GetTSV.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTime**(**args*)
Cryptoki DLL call to CA_GetTime.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTokenCapabilities**(**args*)
Cryptoki DLL call to CA_GetTokenCapabilities.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTokenCertificateInfo**(**args*)
Cryptoki DLL call to CA_GetTokenCertificateInfo.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTokenCertificates**(**args*)
Cryptoki DLL call to CA_GetTokenCertificates.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTokenInsertionCount**(**args*)
Cryptoki DLL call to CA_GetTokenInsertionCount.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTokenObjectHandle**(**args*)
Cryptoki DLL call to CA_GetTokenObjectHandle.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTokenObjectUID**(**args*)
Cryptoki DLL call to CA_GetTokenObjectUID.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTokenPolicies**(**args*)
Cryptoki DLL call to CA_GetTokenPolicies.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTokenStatus**(**args*)
Cryptoki DLL call to CA_GetTokenStatus.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTokenStorageInformation**(**args*)
Cryptoki DLL call to CA_GetTokenStorageInformation.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetTunnelSlotNumber**(**args*)
Cryptoki DLL call to CA_GetTunnelSlotNumber.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetUnauthTokenInsertionStatus**(**args*)
Cryptoki DLL call to CA_GetUnauthTokenInsertionStatus.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetUserContainerName**(**args*)
Cryptoki DLL call to CA_GetUserContainerName.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_GetUserContainerNumber**(**args*)
Cryptoki DLL call to CA_GetUserContainerNumber.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_HAActivateMofN**(**args*)
Cryptoki DLL call to CA_HAActivateMofN.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_HAAnswerLoginChallenge**(**args*)
Cryptoki DLL call to CA_HAAnswerLoginChallenge.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_HAAnswerMofNChallenge**(**args*)
Cryptoki DLL call to CA_HAAnswerMofNChallenge.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_HAGetLoginChallenge**(**args*)
Cryptoki DLL call to CA_HAGetLoginChallenge.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_HAGetMasterPublic**(**args*)
Cryptoki DLL call to CA_HAGetMasterPublic.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_HAInit**(**args*)
Cryptoki DLL call to CA_HAInit.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_HALogin**(**args*)
Cryptoki DLL call to CA_HALogin.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_IndirectLogin**(**args*)
Cryptoki DLL call to CA_IndirectLogin.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_InitAudit**(**args*)
Cryptoki DLL call to CA_InitAudit.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_InitIndirectPIN**(**args*)
Cryptoki DLL call to CA_InitIndirectPIN.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_InitIndirectToken**(**args*)
Cryptoki DLL call to CA_InitIndirectToken.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_InitRolePIN**(**args*)
Cryptoki DLL call to CA_InitRolePIN.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_InitSlotRolePIN**(**args*)
Cryptoki DLL call to CA_InitSlotRolePIN.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_InitializeRemotePEDVector**(**args*)
Cryptoki DLL call to CA_InitializeRemotePEDVector.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_Insert**(**args*)
Cryptoki DLL call to CA_Insert.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM

Returns c_ulong

pycryptoki.cryptoki.**CA_InsertMaskedObject**(**args*)
Cryptoki DLL call to CA_InsertMaskedObject.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_InvokeService**(**args*)
Cryptoki DLL call to CA_InvokeService.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_InvokeServiceAsynch**(**args*)
Cryptoki DLL call to CA_InvokeServiceAsynch.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_InvokeServiceFinal**(**args*)
Cryptoki DLL call to CA_InvokeServiceFinal.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_InvokeServiceInit**(**args*)
Cryptoki DLL call to CA_InvokeServiceInit.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_InvokeServiceSinglePart**(**args*)
Cryptoki DLL call to CA_InvokeServiceSinglePart.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_IsMofNEnabled**(**args*)
Cryptoki DLL call to CA_IsMofNEnabled.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_IsMofNRequired**(**args*)
Cryptoki DLL call to CA_IsMofNRequired.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LKMInitiatorChallenge**(**args*)
Cryptoki DLL call to CA_LKMInitiatorChallenge.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – LP_CK_LKM_TOKEN_ID_S
- **arg6** – LP_CK_LKM_TOKEN_ID_S
- **arg7** – LP_c_ubyte
- **arg8** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LKMInitiatorComplete**(**args*)
Cryptoki DLL call to CA_LKMInitiatorComplete.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

- **arg4** – LP_CK_ATTRIBUTE
- **arg5** – c_ulong
- **arg6** – LP_CK_ATTRIBUTE
- **arg7** – c_ulong
- **arg8** – LP_c_ubyte
- **arg9** – LP_c_ulong
- **arg10** – LP_c_ulong
- **arg11** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LKMReceiverComplete**(**args*)
Cryptoki DLL call to CA_LKMReceiverComplete.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_CK_ATTRIBUTE
- **arg5** – c_ulong
- **arg6** – LP_CK_ATTRIBUTE
- **arg7** – c_ulong
- **arg8** – LP_c_ulong
- **arg9** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LKMReceiverResponse**(**args*)
Cryptoki DLL call to CA_LKMReceiverResponse.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – LP_CK_LKM_TOKEN_ID_S
- **arg6** – LP_c_ubyte
- **arg7** – c_ulong
- **arg8** – LP_c_ubyte
- **arg9** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_ListSecureTokenInit**(**args*)
Cryptoki DLL call to CA_ListSecureTokenInit.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_ListSecureTokenUpdate**(**args*)
Cryptoki DLL call to CA_ListSecureTokenUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LoadEncryptedModule**(**args*)
Cryptoki DLL call to CA_LoadEncryptedModule.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – c_ulong
- **arg7** – LP_c_ubyte
- **arg8** – c_ulong
- **arg9** – LP_c_ubyte
- **arg10** – c_ulong
- **arg11** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LoadModule**(**args*)
Cryptoki DLL call to CA_LoadModule.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte

- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong
- **arg6** – LP_c_ubyte
- **arg7** – c_ulong
- **arg8** – LP_c_ubyte
- **arg9** – c_ulong
- **arg10** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LockClusteredSlot**(**args*)
Cryptoki DLL call to CA_LockClusteredSlot.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LogExportSecret**(**args*)
Cryptoki DLL call to CA_LogExportSecret.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LogExternal**(**args*)
Cryptoki DLL call to CA_LogExternal.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LogGetConfig**(**args*)
Cryptoki DLL call to CA_LogGetConfig.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_LogGetStatus**(**args*)
Cryptoki DLL call to CA_LogGetStatus.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LogImportSecret**(**args*)
Cryptoki DLL call to CA_LogImportSecret.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LogSetConfig**(**args*)
Cryptoki DLL call to CA_LogSetConfig.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – c_ulong
- **arg6** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_LogVerify**(**args*)
Cryptoki DLL call to CA_LogVerify.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_LogVerifyFile**(**args*)
Cryptoki DLL call to CA_LogVerifyFile.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

class pycryptoki.cryptoki.**CA_MOFN_ACTIVATION**

pVector
Structure/Union member

ulVectorLen
Structure/Union member

pycryptoki.cryptoki.**CA_MOFN_ACTIVATION_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CA_MOFN_ACTIVATION

class pycryptoki.cryptoki.**CA_MOFN_GENERATION**

pVector
Structure/Union member

ulVectorLen
Structure/Union member

ulWeight
Structure/Union member

pycryptoki.cryptoki.**CA_MOFN_GENERATION_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CA_MOFN_GENERATION

pycryptoki.cryptoki.**CA_MOFN_STATUS**
alias of pycryptoki.cryptoki.ck_defs.CA_M_OF_N_STATUS

pycryptoki.cryptoki.**CA_MOFN_STATUS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CA_M_OF_N_STATUS

pycryptoki.cryptoki.**CA_MTKGetState**(**args*)
Cryptoki DLL call to CA_MTKGetState.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_MTKResplit**(**args*)
Cryptoki DLL call to CA_MTKResplit.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_MTKRestore**(**args*)
Cryptoki DLL call to CA_MTKRestore.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_MTKSetStorage**(**args*)
Cryptoki DLL call to CA_MTKSetStorage.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_MTKZeroize**(**args*)
Cryptoki DLL call to CA_MTKZeroize.

Parameters **arg1** – c_ulong

Returns c_ulong

class pycryptoki.cryptoki.**CA_M_OF_N_STATUS**

ulFlag
Structure/Union member

ulID
Structure/Union member

ulM
Structure/Union member

ulN
Structure/Union member

ulSecretSize
Structure/Union member

pycryptoki.cryptoki.**CA_ManualKCV**(**args*)
Cryptoki DLL call to CA_ManualKCV.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_ModifyMofN**(**args*)
Cryptoki DLL call to CA_ModifyMofN.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_CA_MOFN_GENERATION
- **arg4** – c_ulong
- **arg5** – c_ulong
- **arg6** – c_void_p

Returns c_ulong

pycryptoki.cryptoki.**CA_ModifyUsageCount**(**args*)
Cryptoki DLL call to CA_ModifyUsageCount.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_MultisignValue**(**args*)
Cryptoki DLL call to CA_MultisignValue.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ulong
- **arg7** – LP_LP_c_ubyte
- **arg8** – LP_c_ulong
- **arg9** – LP_LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_OpenApplicationID**(**args*)
Cryptoki DLL call to CA_OpenApplicationID.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_OpenApplicationIDForContainer**(**args*)
Cryptoki DLL call to CA_OpenApplicationIDForContainer.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_OpenApplicationIDV2**(**args*)
Cryptoki DLL call to CA_OpenApplicationIDV2.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_APPLICATION_ID

Returns c_ulong

pycryptoki.cryptoki.**CA_OpenSecureToken**(**args*)
Cryptoki DLL call to CA_OpenSecureToken.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – c_ulong
- **arg6** – LP_c_ulong
- **arg7** – LP_c_ulong
- **arg8** – c_ulong
- **arg9** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_OpenSession**(**args*)
Cryptoki DLL call to CA_OpenSession.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_void_p
- **arg5** – CFunctionType
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_OpenSessionWithAppID**(**args*)
Cryptoki DLL call to CA_OpenSessionWithAppID.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – c_void_p
- **arg6** – CFunctionType
- **arg7** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_PerformModuleCall**(**args*)
Cryptoki DLL call to CA_PerformModuleCall.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – c_ulong
- **arg7** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_PerformSelfTest**(**args*)
Cryptoki DLL call to CA_PerformSelfTest.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_QueryLicense**(**args*)
Cryptoki DLL call to CA_QueryLicense.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ulong
- **arg7** – LP_c_ubyte

Returns c_ulong

class pycryptoki.cryptoki.**CA_ROLE_STATE**

flags
Structure/Union member

loginAttemptsLeft
Structure/Union member

primaryAuthMech
Structure/Union member

secondaryAuthMech
Structure/Union member

pycryptoki.cryptoki.**CA_ReadCommonStore**(**args*)
Cryptoki DLL call to CA_ReadCommonStore.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_ReplaceFastPathKEK**(**args*)
Cryptoki DLL call to CA_ReplaceFastPathKEK.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_ResetDevice**(**args*)
Cryptoki DLL call to CA_ResetDevice.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_ResetPIN**(**args*)
Cryptoki DLL call to CA_ResetPIN.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_Restart**(**args*)
Cryptoki DLL call to CA_Restart.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_RestartForContainer**(**args*)
Cryptoki DLL call to CA_RestartForContainer.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_RetrieveLicenseList**(**args*)
Cryptoki DLL call to CA_RetrieveLicenseList.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_RoleStateGet**(**args*)
Cryptoki DLL call to CA_RoleStateGet.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_CA_ROLE_STATE

Returns c_ulong

pycryptoki.cryptoki.**CA_SIMExtract**(**args*)
Cryptoki DLL call to CA_SIMExtract.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong
- **arg4** – c_ulong
- **arg5** – c_ulong
- **arg6** – c_ulong
- **arg7** – LP_c_ulong
- **arg8** – LP_LP_c_ubyte
- **arg9** – c_ubyte
- **arg10** – LP_c_ulong
- **arg11** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_SIMInsert**(**args*)
Cryptoki DLL call to CA_SIMInsert.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_LP_c_ubyte

- **arg6** – c_ulong
- **arg7** – LP_c_ubyte
- **arg8** – LP_c_ulong
- **arg9** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SIMMultiSign**(**args*)
Cryptoki DLL call to CA_SIMMultiSign.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – LP_c_ulong
- **arg6** – LP_LP_c_ubyte
- **arg7** – c_ulong
- **arg8** – LP_c_ubyte
- **arg9** – c_ulong
- **arg10** – LP_c_ulong
- **arg11** – LP_LP_c_ubyte
- **arg12** – LP_c_ulong
- **arg13** – LP_LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_STCClearCipherAlgorithm**(**args*)
Cryptoki DLL call to CA_STCClearCipherAlgorithm.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCClearDigestAlgorithm**(**args*)
Cryptoki DLL call to CA_STCClearDigestAlgorithm.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCDeregister**(**args*)
Cryptoki DLL call to CA_STCDeregister.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetAdminPubKey**(**args*)
Cryptoki DLL call to CA_STCGetAdminPubKey.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetChannelID**(**args*)
Cryptoki DLL call to CA_STCGetChannelID.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetCipherAlgorithm**(**args*)
Cryptoki DLL call to CA_STCGetCipherAlgorithm.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetCipherID**(**args*)
Cryptoki DLL call to CA_STCGetCipherID.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetCipherIDs**(**args*)
Cryptoki DLL call to CA_STCGetCipherIDs.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetCipherNameByID**(**args*)
Cryptoki DLL call to CA_STCGetCipherNameByID.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetClientInfo**(**args*)
Cryptoki DLL call to CA_STCGetClientInfo.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ulong
- **arg7** – LP_c_ubyte
- **arg8** – LP_c_ulong
- **arg9** – LP_c_ubyte
- **arg10** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetClientsList**(**args*)
Cryptoki DLL call to CA_STCGetClientsList.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetCurrentKeyLife**(**args*)
Cryptoki DLL call to CA_STCGetCurrentKeyLife.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetDigestAlgorithm**(**args*)
Cryptoki DLL call to CA_STCGetDigestAlgorithm.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetDigestID**(**args*)
Cryptoki DLL call to CA_STCGetDigestID.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetDigestIDs**(**args*)
Cryptoki DLL call to CA_STCGetDigestIDs.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetDigestNameByID**(**args*)
Cryptoki DLL call to CA_STCGetDigestNameByID.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetKeyActivationTimeOut**(**args*)
Cryptoki DLL call to CA_STCGetKeyActivationTimeOut.

Parameters

- **arg1** – c_ulong

- **arg2** – c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetKeyLifeTime**(**args*)
Cryptoki DLL call to CA_STCGetKeyLifeTime.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetMaxSessions**(**args*)
Cryptoki DLL call to CA_STCGetMaxSessions.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetPartPubKey**(**args*)
Cryptoki DLL call to CA_STCGetPartPubKey.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetPubKey**(**args*)
Cryptoki DLL call to CA_STCGetPubKey.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong
- **arg6** – LP_c_ubyte
- **arg7** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetSequenceWindowSize**(**args*)
Cryptoki DLL call to CA_STCGetSequenceWindowSize.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCGetState**(**args*)
Cryptoki DLL call to CA_STCGetState.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_STCIsEnabled**(**args*)
Cryptoki DLL call to CA_STCIsEnabled.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_STCRegister**(**args*)
Cryptoki DLL call to CA_STCRegister.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – c_ulong
- **arg7** – LP_c_ubyte
- **arg8** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCSetCipherAlgorithm**(**args*)
Cryptoki DLL call to CA_STCSetCipherAlgorithm.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCSetDigestAlgorithm**(**args*)
Cryptoki DLL call to CA_STCSetDigestAlgorithm.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCSetKeyActivationTimeOut**(**args*)
Cryptoki DLL call to CA_STCSetKeyActivationTimeOut.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCSetKeyLifeTime**(**args*)
Cryptoki DLL call to CA_STCSetKeyLifeTime.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCSetMaxSessions**(**args*)
Cryptoki DLL call to CA_STCSetMaxSessions.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STCSetSequenceWindowSize**(**args*)
Cryptoki DLL call to CA_STCSetSequenceWindowSize.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STMGetState**(**args*)
Cryptoki DLL call to CA_STMGetState.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_STMToggle**(**args*)
Cryptoki DLL call to CA_STMToggle.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetApplicationID**(**args*)
Cryptoki DLL call to CA_SetApplicationID.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetApplicationIDV2**(**args*)
Cryptoki DLL call to CA_SetApplicationIDV2.

Parameters **arg1** – LP_CK_APPLICATION_ID

Returns c_ulong

pycryptoki.cryptoki.**CA_SetCloningDomain**(**args*)
Cryptoki DLL call to CA_SetCloningDomain.

Parameters

- **arg1** – LP_c_ubyte
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetContainerPolicies**(**args*)
Cryptoki DLL call to CA_SetContainerPolicies.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetContainerPolicy**(**args*)
Cryptoki DLL call to CA_SetContainerPolicy.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetContainerSize**(**args*)
Cryptoki DLL call to CA_SetContainerSize.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetDestructiveHSMPolicies**(**args*)
Cryptoki DLL call to CA_SetDestructiveHSMPolicies.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetDestructiveHSMPolicy**(**args*)
Cryptoki DLL call to CA_SetDestructiveHSMPolicy.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetExtendedTPV**(**args*)
Cryptoki DLL call to CA_SetExtendedTPV.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetHSMPolicies**(**args*)
Cryptoki DLL call to CA_SetHSMPolicies.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetHSMPolicy**(**args*)
Cryptoki DLL call to CA_SetHSMPolicy.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetKCV**(**args*)
Cryptoki DLL call to CA_SetKCV.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetLKCV**(**args*)
Cryptoki DLL call to CA_SetLKCV.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetMofN**(**args*)
Cryptoki DLL call to CA_SetMofN.

Parameters **arg1** – c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**CA_SetPedId**(**args*)
Cryptoki DLL call to CA_SetPedId.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetRDK**(**args*)
Cryptoki DLL call to CA_SetRDK.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetTPV**(**args*)
Cryptoki DLL call to CA_SetTPV.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetTokenCertificateSignature**(**args*)
Cryptoki DLL call to CA_SetTokenCertificateSignature.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_CK_ATTRIBUTE
- **arg5** – c_ulong
- **arg6** – LP_c_ubyte
- **arg7** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetTokenPolicies**(**args*)
Cryptoki DLL call to CA_SetTokenPolicies.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SetUserContainerName**(**args*)
Cryptoki DLL call to CA_SetUserContainerName.

Parameters

- **arg1** – c_ulong

- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SpRawRead**(**args*)
Cryptoki DLL call to CA_SpRawRead.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SpRawWrite**(**args*)
Cryptoki DLL call to CA_SpRawWrite.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_SwitchSecondarySlot**(**args*)
Cryptoki DLL call to CA_SwitchSecondarySlot.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_TamperClear**(**args*)
Cryptoki DLL call to CA_TamperClear.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_TimeSync**(**args*)
Cryptoki DLL call to CA_TimeSync.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_TokenDelete**(**args*)
Cryptoki DLL call to CA_TokenDelete.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_TokenInsert**(**args*)
Cryptoki DLL call to CA_TokenInsert.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CT_Token
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_TokenInsertNoAuth**(**args*)
Cryptoki DLL call to CA_TokenInsertNoAuth.

Parameters

- **arg1** – LP_CT_Token
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_TokenZeroize**(**args*)
Cryptoki DLL call to CA_TokenZeroize.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_UnloadModule**(**args*)
Cryptoki DLL call to CA_UnloadModule.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_UnlockClusteredSlot**(**args*)
Cryptoki DLL call to CA_UnlockClusteredSlot.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_WaitForSlotEvent**(**args*)
Cryptoki DLL call to CA_WaitForSlotEvent.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong
- **arg4** – c_void_p

Returns c_ulong

pycryptoki.cryptoki.**CA_WriteCommonStore**(**args*)
Cryptoki DLL call to CA_WriteCommonStore.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CA_Zeroize**(**args*)
Cryptoki DLL call to CA_Zeroize.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**CKA_SIM_AUTH_FORM**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CKCA_MODULE_ID**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CKCA_MODULE_ID_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

class pycryptoki.cryptoki.**CKCA_MODULE_INFO**

developerName
Structure/Union member

moduleDescription
Structure/Union member

moduleVersion
Structure/Union member

ulModuleSize
Structure/Union member

pycryptoki.cryptoki.**CKCA_MODULE_INFO_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CKCA_MODULE_INFO

class pycryptoki.cryptoki.**CK_AES_CBC_ENCRYPT_DATA_PARAMS**

iv
Structure/Union member

length
Structure/Union member

pData
Structure/Union member

pycryptoki.cryptoki.**CK_AES_CBC_ENCRYPT_DATA_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_AES_CBC_ENCRYPT_DATA_PARAMS

class pycryptoki.cryptoki.**CK_AES_CBC_PAD_EXTRACT_PARAMS**

ctxID
Structure/Union member

pBuffer
Structure/Union member

pbFileName
Structure/Union member

pedId
Structure/Union member

pulBufferLen
Structure/Union member

ulDeleteAfterExtract
Structure/Union member

ulHandle
Structure/Union member

ulStorage
Structure/Union member

ulType
Structure/Union member

pycryptoki.cryptoki.**CK_AES_CBC_PAD_EXTRACT_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_AES_CBC_PAD_EXTRACT_PARAMS

class pycryptoki.cryptoki.**CK_AES_CBC_PAD_INSERT_PARAMS**

ctxID
Structure/Union member

pBuffer
Structure/Union member

pbFileName
Structure/Union member

pedId
Structure/Union member

pulHandle
Structure/Union member

pulType
Structure/Union member

ulBufferLen
Structure/Union member

ulContainerState
Structure/Union member

ulStorage
Structure/Union member

ulStorageType
Structure/Union member

pycryptoki.cryptoki.**CK_AES_CBC_PAD_INSERT_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_AES_CBC_PAD_INSERT_PARAMS

class pycryptoki.cryptoki.**CK_AES_CTR_PARAMS**

cb
Structure/Union member

ulCounterBits
Structure/Union member

pycryptoki.cryptoki.**CK_AES_CTR_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_AES_CTR_PARAMS

class pycryptoki.cryptoki.**CK_AES_GCM_PARAMS**

pAAD
Structure/Union member

pIv
Structure/Union member

ulAADLen
Structure/Union member

ulIvBits
Structure/Union member

ulIvLen
Structure/Union member

ulTagBits
Structure/Union member

pycryptoki.cryptoki.**CK_AES_GCM_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.CK_AES_GCM_PARAMS

pycryptoki.cryptoki.**CK_AES_GMAC_PARAMS**
alias of pycryptoki.cryptoki.ck_defs.CK_AES_GCM_PARAMS

pycryptoki.cryptoki.**CK_AES_GMAC_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_AES_GCM_PARAMS

class pycryptoki.cryptoki.**CK_AES_XTS_PARAMS**

cb
Structure/Union member

hTweakKey
Structure/Union member

pycryptoki.cryptoki.**CK_AES_XTS_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_AES_XTS_PARAMS

class pycryptoki.cryptoki.**CK_ARIA_CBC_ENCRYPT_DATA_PARAMS**

iv
Structure/Union member

length
Structure/Union member

pData
Structure/Union member

pycryptoki.cryptoki.**CK_ARIA_CBC_ENCRYPT_DATA_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_ARIA_CBC_ENCRYPT_DATA_PARAMS

pycryptoki.cryptoki.**CK_ARIA_CTR_PARAMS**
alias of pycryptoki.cryptoki.ck_defs.CK_AES_CTR_PARAMS

pycryptoki.cryptoki.**CK_ARIA_CTR_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_AES_CTR_PARAMS

class pycryptoki.cryptoki.**CK_ATTRIBUTE**

pValue
Structure/Union member

type
Structure/Union member

usValueLen
Structure/Union member

pycryptoki.cryptoki.**CK_ATTRIBUTE_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_ATTRIBUTE

pycryptoki.cryptoki.**CK_ATTRIBUTE_TYPE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_BBOOL**
alias of ctypes.c_ubyte

pycryptoki.cryptoki.**CK_BYTE**
alias of ctypes.c_ubyte

pycryptoki.cryptoki.**CK_BYTE_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ubyte

class pycryptoki.cryptoki.**CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS**

iv
Structure/Union member

length
Structure/Union member

pData
Structure/Union member

pycryptoki.cryptoki.**CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS

class pycryptoki.cryptoki.**CK_CAMELLIA_CTR_PARAMS**

cb
Structure/Union member

ulCounterBits
Structure/Union member

pycryptoki.cryptoki.**CK_CAMELLIA_CTR_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_CAMELLIA_CTR_PARAMS

pycryptoki.cryptoki.**CK_CERTIFICATE_TYPE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_CHAR**
alias of ctypes.c_ubyte

pycryptoki.cryptoki.**CK_CHAR_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ubyte

class pycryptoki.cryptoki.**CK_CLUSTER_STATE**

bMembers
Structure/Union member

ulMemberStatus
Structure/Union member

pycryptoki.cryptoki.**CK_CLUSTER_STATE_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_CLUSTER_STATE

class pycryptoki.cryptoki.**CK_CMS_SIG_PARAMS**

certificateHandle
Structure/Union member

pContentType
Structure/Union member

pDigestMechanism
Structure/Union member

pRequestedAttributes
Structure/Union member

pRequiredAttributes
Structure/Union member

pSigningMechanism
Structure/Union member

ulRequestedAttributesLen
Structure/Union member

ulRequiredAttributesLen
Structure/Union member

pycryptoki.cryptoki.**CK_CMS_SIG_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_CMS_SIG_PARAMS

pycryptoki.cryptoki.**CK_CREATEMUTEX**
alias of ctypes.CFUNCTYPE.<locals>.CFunctionType

class pycryptoki.cryptoki.**CK_DATE**

day
Structure/Union member

month
Structure/Union member

year
Structure/Union member

pycryptoki.cryptoki.**CK_DESTROYMUTEX**
alias of ctypes.CFUNCTYPE.<locals>.CFunctionType

class pycryptoki.cryptoki.**CK_DES_CBC_ENCRYPT_DATA_PARAMS**

iv
Structure/Union member

length
Structure/Union member

pData
Structure/Union member

pycryptoki.cryptoki.**CK_DES_CBC_ENCRYPT_DATA_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_DES_CBC_ENCRYPT_DATA_PARAMS

class pycryptoki.cryptoki.**CK_DES_CTR_PARAMS**

cb
Structure/Union member

ulCounterBits
Structure/Union member

pycryptoki.cryptoki.**CK_DES_CTR_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_DES_CTR_PARAMS

class pycryptoki.cryptoki.**CK_ECDH1_DERIVE_PARAMS**

kdf
Structure/Union member

pPublicData
Structure/Union member

pSharedData
Structure/Union member

ulPublicDataLen
Structure/Union member

ulSharedDataLen
Structure/Union member

pycryptoki.cryptoki.**CK_ECDH1_DERIVE_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_ECDH1_DERIVE_PARAMS

class pycryptoki.cryptoki.**CK_ECDH2_DERIVE_PARAMS**

hPrivateData
Structure/Union member

kdf
Structure/Union member

pPublicData
Structure/Union member

pPublicData2
Structure/Union member

pSharedData
Structure/Union member

ulPrivateDataLen
Structure/Union member

ulPublicDataLen
Structure/Union member

ulPublicDataLen2
Structure/Union member

ulSharedDataLen
Structure/Union member

pycryptoki.cryptoki.**CK_ECDH2_DERIVE_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_ECDH2_DERIVE_PARAMS

class pycryptoki.cryptoki.**CK_ECIES_PARAMS**

dhPrimitive
Structure/Union member

encScheme
Structure/Union member

kdf
Structure/Union member

macScheme
Structure/Union member

pSharedData1
Structure/Union member

pSharedData2
Structure/Union member

ulEncKeyLenInBits
Structure/Union member

ulMacKeyLenInBits
Structure/Union member

ulMacLenInBits
Structure/Union member

ulSharedDataLen1
Structure/Union member

ulSharedDataLen2
Structure/Union member

pycryptoki.cryptoki.**CK_ECIES_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_ECIES_PARAMS

class pycryptoki.cryptoki.**CK_ECMQV_DERIVE_PARAMS**

hPrivateData
Structure/Union member

kdf
Structure/Union member

pPublicData
Structure/Union member

pPublicData2
Structure/Union member

pSharedData
Structure/Union member

publicKey
Structure/Union member

ulPrivateDataLen
Structure/Union member

ulPublicDataLen
Structure/Union member

ulPublicDataLen2
Structure/Union member

ulSharedDataLen
Structure/Union member

pycryptoki.cryptoki.**CK_ECMQV_DERIVE_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_ECMQV_DERIVE_PARAMS

pycryptoki.cryptoki.**CK_EC_DH_PRIMITIVE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_EC_ENC_SCHEME**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_EC_KDF_TYPE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_EC_MAC_SCHEME**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_EXTRACT_PARAMS**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_EXTRACT_PARAMS_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

pycryptoki.cryptoki.**CK_FLAGS**
alias of ctypes.c_ulong

class pycryptoki.cryptoki.**CK_FUNCTION_LIST**

C_CancelFunction
Structure/Union member

C_CloseAllSessions
Structure/Union member

C_CloseSession
Structure/Union member

C_CopyObject
Structure/Union member

C_CreateObject
Structure/Union member

C_Decrypt
Structure/Union member

C_DecryptDigestUpdate
Structure/Union member

C_DecryptFinal
Structure/Union member

C_DecryptInit
Structure/Union member

C_DecryptUpdate
Structure/Union member

C_DecryptVerifyUpdate
Structure/Union member

C_DeriveKey
Structure/Union member

C_DestroyObject
Structure/Union member

C_Digest
Structure/Union member

C_DigestEncryptUpdate
Structure/Union member

C_DigestFinal
Structure/Union member

C_DigestInit
Structure/Union member

C_DigestKey
Structure/Union member

C_DigestUpdate
Structure/Union member

C_Encrypt
Structure/Union member

C_EncryptFinal
Structure/Union member

C_EncryptInit
Structure/Union member

C_EncryptUpdate
Structure/Union member

C_Finalize
Structure/Union member

C_FindObjects
Structure/Union member

C_FindObjectsFinal
Structure/Union member

C_FindObjectsInit
Structure/Union member

C_GenerateKey
Structure/Union member

C_GenerateKeyPair
Structure/Union member

C_GenerateRandom
Structure/Union member

C_GetAttributeValue
Structure/Union member

C_GetFunctionList
Structure/Union member

C_GetFunctionStatus
Structure/Union member

C_GetInfo
Structure/Union member

C_GetMechanismInfo
Structure/Union member

C_GetMechanismList
Structure/Union member

C_GetObjectSize
Structure/Union member

C_GetOperationState
Structure/Union member

C_GetSessionInfo
Structure/Union member

C_GetSlotInfo
Structure/Union member

C_GetSlotList
Structure/Union member

C_GetTokenInfo
Structure/Union member

C_InitPIN
Structure/Union member

C_InitToken
Structure/Union member

C_Initialize
Structure/Union member

C_Login
Structure/Union member

C_Logout
Structure/Union member

C_OpenSession
Structure/Union member

C_SeedRandom
Structure/Union member

C_SetAttributeValue
Structure/Union member

C_SetOperationState
Structure/Union member

C_SetPIN
Structure/Union member

C_Sign
Structure/Union member

C_SignEncryptUpdate
Structure/Union member

C_SignFinal
Structure/Union member

C_SignInit
Structure/Union member

C_SignRecover
Structure/Union member

C_SignRecoverInit
Structure/Union member

C_SignUpdate
Structure/Union member

C_UnwrapKey
Structure/Union member

C_Verify
Structure/Union member

C_VerifyFinal
Structure/Union member

C_VerifyInit
Structure/Union member

C_VerifyRecover
Structure/Union member

C_VerifyRecoverInit
Structure/Union member

C_VerifyUpdate
Structure/Union member

C_WaitForSlotEvent
Structure/Union member

C_WrapKey
Structure/Union member

version
Structure/Union member

pycryptoki.cryptoki.**CK_FUNCTION_LIST_PTR**
alias of pycryptoki.cryptoki._ck_func_list.LP_CK_FUNCTION_LIST

pycryptoki.cryptoki.**CK_FUNCTION_LIST_PTR_PTR**
alias of pycryptoki.cryptoki._ck_func_list.LP_LP_CK_FUNCTION_LIST

pycryptoki.cryptoki.**CK_GetTotalOperations**
alias of ctypes.CFUNCTYPE.<locals>.CFunctionType

class pycryptoki.cryptoki.**CK_HA_MEMBER**

memberSerial
Structure/Union member

memberStatus
Structure/Union member

pycryptoki.cryptoki.**CK_HA_MEMBER_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_HA_MEMBER

pycryptoki.cryptoki.**CK_HA_STATE_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_HA_STATUS

class pycryptoki.cryptoki.**CK_HA_STATUS**

groupSerial
Structure/Union member

listSize
Structure/Union member

memberList
Structure/Union member

pycryptoki.cryptoki.**CK_HW_FEATURE_TYPE**
alias of ctypes.c_ulong

class pycryptoki.cryptoki.**CK_INFO**

cryptokiVersion
Structure/Union member

flags
Structure/Union member

libraryDescription
Structure/Union member

libraryVersion
Structure/Union member

manufacturerID
Structure/Union member

pycryptoki.cryptoki.**CK_INFO_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_INFO

pycryptoki.cryptoki.**CK_KDF_PRF_ENCODING_SCHEME**
alias of ctypes.c_ulong

class pycryptoki.cryptoki.**CK_KDF_PRF_PARAMS**

pContext
Structure/Union member

pLabel
Structure/Union member

prfType
Structure/Union member

ulContextLen
Structure/Union member

ulCounter
Structure/Union member

ulEncodingScheme
Structure/Union member

ulLabelLen
Structure/Union member

pycryptoki.cryptoki.**CK_KDF_PRF_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_KDF_PRF_PARAMS

pycryptoki.cryptoki.**CK_KDF_PRF_TYPE**
alias of ctypes.c_ulong

class pycryptoki.cryptoki.**CK_KEA_DERIVE_PARAMS**

isSender
Structure/Union member

pPublicData
Structure/Union member

pRandomA
Structure/Union member

pRandomB
Structure/Union member

ulPublicDataLen
Structure/Union member

ulRandomLen
Structure/Union member

pycryptoki.cryptoki.**CK_KEA_DERIVE_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_KEA_DERIVE_PARAMS

class pycryptoki.cryptoki.**CK_KEY_DERIVATION_STRING_DATA**

pData
Structure/Union member

ulLen
Structure/Union member

pycryptoki.cryptoki.**CK_KEY_DERIVATION_STRING_DATA_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_KEY_DERIVATION_STRING_DATA

pycryptoki.cryptoki.**CK_KEY_TYPE**
alias of ctypes.c_ulong

class pycryptoki.cryptoki.**CK_KEY_WRAP_SET_OAEP_PARAMS**

bBC
Structure/Union member

pX
Structure/Union member

ulXLen
Structure/Union member

pycryptoki.cryptoki.**CK_KEY_WRAP_SET_OAEP_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_KEY_WRAP_SET_OAEP_PARAMS

class pycryptoki.cryptoki.**CK_KIP_PARAMS**

hKey
Structure/Union member

pMechanism
Structure/Union member

pSeed
Structure/Union member

ulSeedLen
Structure/Union member

pycryptoki.cryptoki.**CK_KIP_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_KIP_PARAMS

pycryptoki.cryptoki.**CK_LKM_TOKEN_ID**
alias of pycryptoki.cryptoki.ck_defs.CK_LKM_TOKEN_ID_S

pycryptoki.cryptoki.**CK_LKM_TOKEN_ID_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_LKM_TOKEN_ID_S

class pycryptoki.cryptoki.**CK_LKM_TOKEN_ID_S**

id
Structure/Union member

pycryptoki.cryptoki.**CK_LOCKMUTEX**
alias of ctypes.CFUNCTYPE.<locals>.CFunctionType

pycryptoki.cryptoki.**CK_LONG**
alias of ctypes.c_long

pycryptoki.cryptoki.**CK_MAC_GENERAL_PARAMS**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_MAC_GENERAL_PARAMS_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

class pycryptoki.cryptoki.**CK_MECHANISM**

mechanism
Structure/Union member

pParameter
Structure/Union member

usParameterLen
Structure/Union member

class pycryptoki.cryptoki.**CK_MECHANISM_INFO**

flags
Structure/Union member

ulMaxKeySize
Structure/Union member

ulMinKeySize
Structure/Union member

pycryptoki.cryptoki.**CK_MECHANISM_INFO_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_MECHANISM_INFO

pycryptoki.cryptoki.**CK_MECHANISM_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_MECHANISM

pycryptoki.cryptoki.**CK_MECHANISM_TYPE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_MECHANISM_TYPE_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

pycryptoki.cryptoki.**CK_NOTIFICATION**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_NOTIFY**
alias of ctypes.CFUNCTYPE.<locals>.CFunctionType

pycryptoki.cryptoki.**CK_OBJECT_CLASS**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_OBJECT_CLASS_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

pycryptoki.cryptoki.**CK_OBJECT_HANDLE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_OBJECT_HANDLE_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

class pycryptoki.cryptoki.**CK_OTP_PARAM**

pValue
Structure/Union member

type
Structure/Union member

usValueLen
Structure/Union member

class pycryptoki.cryptoki.**CK_OTP_PARAMS**

pParams
Structure/Union member

ulCount
Structure/Union member

pycryptoki.cryptoki.**CK_OTP_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_OTP_PARAMS

pycryptoki.cryptoki.**CK_OTP_PARAM_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_OTP_PARAM

pycryptoki.cryptoki.**CK_OTP_PARAM_TYPE**
alias of ctypes.c_ulong

class pycryptoki.cryptoki.**CK_OTP_SIGNATURE_INFO**

pParams
Structure/Union member

ulCount
Structure/Union member

pycryptoki.cryptoki.**CK_OTP_SIGNATURE_INFO_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_OTP_SIGNATURE_INFO

pycryptoki.cryptoki.**CK_PARAM_TYPE**
alias of ctypes.c_ulong

class pycryptoki.cryptoki.**CK_PBE_PARAMS**

pInitVector
Structure/Union member

pPassword
Structure/Union member

pSalt
Structure/Union member

usIteration
Structure/Union member

usPasswordLen
Structure/Union member

usSaltLen
Structure/Union member

pycryptoki.cryptoki.**CK_PBE_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_PBE_PARAMS

class pycryptoki.cryptoki.**CK_PKCS5_PBKD2_PARAMS**

iterations
Structure/Union member

pPassword
Structure/Union member

pPrfData
Structure/Union member

pSaltSourceData
Structure/Union member

prf
Structure/Union member

saltSource
Structure/Union member

ulPrfDataLen
Structure/Union member

ulSaltSourceDataLen
Structure/Union member

usPasswordLen
Structure/Union member

pycryptoki.cryptoki.**CK_PKCS5_PBKD2_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_PKCS5_PBKD2_PARAMS

pycryptoki.cryptoki.**CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

pycryptoki.cryptoki.**CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

pycryptoki.cryptoki.**CK_PRF_KDF_PARAMS**
alias of pycryptoki.cryptoki.ck_defs.CK_KDF_PRF_PARAMS

class pycryptoki.cryptoki.**CK_RC2_CBC_PARAMS**

iv
Structure/Union member

usEffectiveBits
Structure/Union member

pycryptoki.cryptoki.**CK_RC2_CBC_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_RC2_CBC_PARAMS

class pycryptoki.cryptoki.**CK_RC2_MAC_GENERAL_PARAMS**

ulMacLength
Structure/Union member

usEffectiveBits
Structure/Union member

pycryptoki.cryptoki.**CK_RC2_MAC_GENERAL_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_RC2_MAC_GENERAL_PARAMS

pycryptoki.cryptoki.**CK_RC2_PARAMS**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_RC2_PARAMS_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

class pycryptoki.cryptoki.**CK_RC5_CBC_PARAMS**

pIv
Structure/Union member

ulIvLen
Structure/Union member

ulRounds
Structure/Union member

ulWordsize
Structure/Union member

pycryptoki.cryptoki.**CK_RC5_CBC_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_RC5_CBC_PARAMS

class pycryptoki.cryptoki.**CK_RC5_MAC_GENERAL_PARAMS**

ulMacLength
Structure/Union member

ulRounds
Structure/Union member

ulWordsize
Structure/Union member

pycryptoki.cryptoki.**CK_RC5_MAC_GENERAL_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_RC5_MAC_GENERAL_PARAMS

class pycryptoki.cryptoki.**CK_RC5_PARAMS**

ulRounds
Structure/Union member

ulWordsize
Structure/Union member

pycryptoki.cryptoki.**CK_RC5_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_RC5_PARAMS

pycryptoki.cryptoki.**CK_RSA_PKCS_MGF_TYPE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_RSA_PKCS_MGF_TYPE_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

class pycryptoki.cryptoki.**CK_RSA_PKCS_OAEP_PARAMS**

hashAlg
Structure/Union member

mgf
Structure/Union member

pSourceData
Structure/Union member

source
Structure/Union member

ulSourceDataLen
Structure/Union member

pycryptoki.cryptoki.**CK_RSA_PKCS_OAEP_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_RSA_PKCS_OAEP_PARAMS

pycryptoki.cryptoki.**CK_RSA_PKCS_OAEP_SOURCE_TYPE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_RSA_PKCS_OAEP_SOURCE_TYPE_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

class pycryptoki.cryptoki.**CK_RSA_PKCS_PSS_PARAMS**

hashAlg
Structure/Union member

mgf
Structure/Union member

usSaltLen
Structure/Union member

pycryptoki.cryptoki.**CK_RSA_PKCS_PSS_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_RSA_PKCS_PSS_PARAMS

pycryptoki.cryptoki.**CK_RV**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_ResetTotalOperations**
alias of ctypes.CFUNCTYPE.<locals>.CFunctionType

pycryptoki.cryptoki.**CK_SEED_CTR_PARAMS**
alias of pycryptoki.cryptoki.ck_defs.CK_AES_CTR_PARAMS

pycryptoki.cryptoki.**CK_SEED_CTR_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_AES_CTR_PARAMS

pycryptoki.cryptoki.**CK_SESSION_HANDLE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_SESSION_HANDLE_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

class pycryptoki.cryptoki.**CK_SESSION_INFO**

flags
Structure/Union member

slotID
Structure/Union member

state
Structure/Union member

usDeviceError
Structure/Union member

pycryptoki.cryptoki.**CK_SESSION_INFO_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_SESSION_INFO

class pycryptoki.cryptoki.**CK_SFNT_CA_FUNCTION_LIST**

CA_ActivateMofN
Structure/Union member

CA_AuthorizeKey
Structure/Union member

CA_CapabilityUpdate
Structure/Union member

CA_CheckOperationState
Structure/Union member

CA_ChoosePrimarySlot
Structure/Union member

CA_ChooseSecondarySlot
Structure/Union member

CA_CloneAllObjectsToSession
Structure/Union member

CA_CloneAsSource
Structure/Union member

CA_CloneAsTarget
Structure/Union member

CA_CloneAsTargetInit
Structure/Union member

CA_CloneModifyMofN
Structure/Union member

CA_CloneMofN
Structure/Union member

CA_CloneObject
Structure/Union member

CA_CloneObjectToAllSessions
Structure/Union member

CA_ClonePrivateKey
Structure/Union member

CA_CloseAllSecondarySessions
Structure/Union member

CA_CloseApplicationID
Structure/Union member

CA_CloseApplicationIDForContainer
Structure/Union member

CA_CloseSecondarySession
Structure/Union member

CA_CloseSecureToken
Structure/Union member

CA_ConfigureRemotePED
Structure/Union member

CA_CreateContainer
Structure/Union member

CA_CreateContainerLoginChallenge
Structure/Union member

CA_CreateLoginChallenge
Structure/Union member

CA_Deactivate
Structure/Union member

CA_DeactivateMofN
Structure/Union member

CA_DeleteContainer
Structure/Union member

CA_DeleteContainerWithHandle
Structure/Union member

CA_DeleteRemotePEDVector
Structure/Union member

CA_DescribeUtilizationBinId
Structure/Union member

CA_DestroyMultipleObjects
Structure/Union member

CA_DisableUnauthTokenInsertion
Structure/Union member

CA_DismantleRemotePED
Structure/Union member

CA_DuplicateMofN
Structure/Union member

CA_EnableUnauthTokenInsertion
Structure/Union member

CA_EncodeECChar2Params
Structure/Union member

CA_EncodeECParamsFromFile
Structure/Union member

CA_EncodeECPrimeParams
Structure/Union member

CA_Extract
Structure/Union member

CA_ExtractMaskedObject
Structure/Union member

CA_FactoryReset
Structure/Union member

CA_FindAdminSlotForSlot
Structure/Union member

CA_FirmwareRollback
Structure/Union member

CA_FirmwareUpdate
Structure/Union member

CA_GenerateCloneableMofN
Structure/Union member

CA_GenerateCloningKEV
Structure/Union member

CA_GenerateMofN
Structure/Union member

CA_GenerateTokenKeys
Structure/Union member

CA_GetCVFirmwareVersion
Structure/Union member

CA_GetClusterState
Structure/Union member

CA_GetConfigurationElementDescription
Structure/Union member

CA_GetContainerCapabilitySet
Structure/Union member

CA_GetContainerCapabilitySetting
Structure/Union member

CA_GetContainerList
Structure/Union member

CA_GetContainerName
Structure/Union member

CA_GetContainerPolicySet
Structure/Union member

CA_GetContainerPolicySetting
Structure/Union member

CA_GetContainerStatus
Structure/Union member

CA_GetContainerStorageInformation
Structure/Union member

CA_GetExtendedTPV
Structure/Union member

CA_GetFPV
Structure/Union member

CA_GetFirmwareVersion
Structure/Union member

CA_GetFunctionList
Structure/Union member

CA_GetHAState
Structure/Union member

CA_GetHSMCapabilitySet
Structure/Union member

CA_GetHSMCapabilitySetting
Structure/Union member

CA_GetHSMPolicySet
Structure/Union member

CA_GetHSMPolicySetting
Structure/Union member

CA_GetHSMStats
Structure/Union member

CA_GetHSMStorageInformation
Structure/Union member

CA_GetModuleInfo
Structure/Union member

CA_GetModuleList
Structure/Union member

CA_GetMofNStatus
Structure/Union member

CA_GetNumberOfAllowedContainers
Structure/Union member

CA_GetObjectHandle
Structure/Union member

CA_GetObjectUID
Structure/Union member

CA_GetPedId
Structure/Union member

CA_GetPrimarySlot
Structure/Union member

CA_GetRemotePEDVectorStatus
Structure/Union member

CA_GetRollbackFirmwareVersion
Structure/Union member

CA_GetSecondarySlot
Structure/Union member

CA_GetSecureElementMeta
Structure/Union member

CA_GetServerInstanceBySlotID
Structure/Union member

CA_GetSessionInfo
Structure/Union member

CA_GetSlotIdForContainer
Structure/Union member

CA_GetSlotIdForPhysicalSlot
Structure/Union member

CA_GetSlotListFromServerInstance
Structure/Union member

CA_GetTPV
Structure/Union member

CA_GetTSV
Structure/Union member

CA_GetTime
Structure/Union member

CA_GetTokenCapabilities
Structure/Union member

CA_GetTokenCertificateInfo
Structure/Union member

CA_GetTokenCertificates
Structure/Union member

CA_GetTokenInsertionCount
Structure/Union member

CA_GetTokenObjectHandle
Structure/Union member

CA_GetTokenObjectUID
Structure/Union member

CA_GetTokenPolicies
Structure/Union member

CA_GetTokenStatus
Structure/Union member

CA_GetTokenStorageInformation
Structure/Union member

CA_GetTunnelSlotNumber
Structure/Union member

CA_GetUnauthTokenInsertionStatus
Structure/Union member

CA_GetUserContainerName
Structure/Union member

CA_GetUserContainerNumber
Structure/Union member

CA_HAActivateMofN
Structure/Union member

CA_HAAnswerLoginChallenge
Structure/Union member

CA_HAAnswerMofNChallenge
Structure/Union member

CA_HAGetLoginChallenge
Structure/Union member

CA_HAGetMasterPublic
Structure/Union member

CA_HAInit
Structure/Union member

CA_HALogin
Structure/Union member

CA_IndirectLogin
Structure/Union member

CA_InitAudit
Structure/Union member

CA_InitIndirectPIN
Structure/Union member

CA_InitIndirectToken
Structure/Union member

CA_InitRolePIN
Structure/Union member

CA_InitSlotRolePIN
Structure/Union member

CA_InitializeRemotePEDVector
Structure/Union member

CA_Insert
Structure/Union member

CA_InsertMaskedObject
Structure/Union member

CA_InvokeService
Structure/Union member

CA_InvokeServiceAsynch
Structure/Union member

CA_InvokeServiceFinal
Structure/Union member

CA_InvokeServiceInit
Structure/Union member

CA_InvokeServiceSinglePart
Structure/Union member

CA_IsMofNEnabled
Structure/Union member

CA_IsMofNRequired
Structure/Union member

CA_LKMInitiatorChallenge
Structure/Union member

CA_LKMInitiatorComplete
Structure/Union member

CA_LKMReceiverComplete
Structure/Union member

CA_LKMReceiverResponse
Structure/Union member

CA_ListSecureTokenInit
Structure/Union member

CA_ListSecureTokenUpdate
Structure/Union member

CA_LoadEncryptedModule
Structure/Union member

CA_LoadModule
Structure/Union member

CA_LockClusteredSlot
Structure/Union member

CA_LogExportSecret
Structure/Union member

CA_LogExternal
Structure/Union member

CA_LogGetConfig
Structure/Union member

CA_LogGetStatus
Structure/Union member

CA_LogImportSecret
Structure/Union member

CA_LogSetConfig
Structure/Union member

CA_LogVerify
Structure/Union member

CA_LogVerifyFile
Structure/Union member

CA_MTKGetState
Structure/Union member

CA_MTKResplit
Structure/Union member

CA_MTKRestore
Structure/Union member

CA_MTKSetStorage
Structure/Union member

CA_MTKZeroize
Structure/Union member

CA_ManualKCV
Structure/Union member

CA_ModifyMofN
Structure/Union member

CA_ModifyUsageCount
Structure/Union member

CA_MultisignValue
Structure/Union member

CA_OpenApplicationID
Structure/Union member

CA_OpenApplicationIDForContainer
Structure/Union member

CA_OpenSecureToken
Structure/Union member

CA_OpenSession
Structure/Union member

CA_OpenSessionWithAppID
Structure/Union member

CA_PerformModuleCall
Structure/Union member

CA_PerformSelfTest
Structure/Union member

CA_QueryLicense
Structure/Union member

CA_ReadAllUtilizationCounters
Structure/Union member

CA_ReadAndResetUtilizationMetrics
Structure/Union member

CA_ReadCommonStore
Structure/Union member

CA_ReadUtilizationMetrics
Structure/Union member

CA_ReplaceFastPathKEK
Structure/Union member

CA_ResetDevice
Structure/Union member

CA_ResetPIN
Structure/Union member

CA_Restart
Structure/Union member

CA_RestartForContainer
Structure/Union member

CA_RetrieveLicenseList
Structure/Union member

CA_RoleStateGet
Structure/Union member

CA_SIMExtract
Structure/Union member

CA_SIMInsert
Structure/Union member

CA_SIMMultiSign
Structure/Union member

CA_STCClearCipherAlgorithm
Structure/Union member

CA_STCClearDigestAlgorithm
Structure/Union member

CA_STCDeregister
Structure/Union member

CA_STCGetAdminPubKey
Structure/Union member

CA_STCGetChannelID
Structure/Union member

CA_STCGetCipherAlgorithm
Structure/Union member

CA_STCGetCipherID
Structure/Union member

CA_STCGetCipherIDs
Structure/Union member

CA_STCGetCipherNameByID
Structure/Union member

CA_STCGetClientInfo
Structure/Union member

CA_STCGetClientsList
Structure/Union member

CA_STCGetCurrentKeyLife
Structure/Union member

CA_STCGetDigestAlgorithm
Structure/Union member

CA_STCGetDigestID
Structure/Union member

CA_STCGetDigestIDs
Structure/Union member

CA_STCGetDigestNameByID
Structure/Union member

CA_STCGetKeyActivationTimeOut
Structure/Union member

CA_STCGetKeyLifeTime
Structure/Union member

CA_STCGetMaxSessions
Structure/Union member

CA_STCGetPartPubKey
Structure/Union member

CA_STCGetPubKey
Structure/Union member

CA_STCGetSequenceWindowSize
Structure/Union member

CA_STCGetState
Structure/Union member

CA_STCIsEnabled
Structure/Union member

CA_STCRegister
Structure/Union member

CA_STCSetCipherAlgorithm
Structure/Union member

CA_STCSetDigestAlgorithm
Structure/Union member

CA_STCSetKeyActivationTimeOut
Structure/Union member

CA_STCSetKeyLifeTime
Structure/Union member

CA_STCSetMaxSessions
Structure/Union member

CA_STCSetSequenceWindowSize
Structure/Union member

CA_STMGetState
Structure/Union member

CA_STMToggle
Structure/Union member

CA_SetApplicationID
Structure/Union member

CA_SetAuthorizationData
Structure/Union member

CA_SetCloningDomain
Structure/Union member

CA_SetContainerPolicies
Structure/Union member

CA_SetContainerPolicy
Structure/Union member

CA_SetContainerSize
Structure/Union member

CA_SetDestructiveHSMPolicies
Structure/Union member

CA_SetDestructiveHSMPolicy
Structure/Union member

CA_SetExtendedTPV
Structure/Union member

CA_SetHSMPolicies
Structure/Union member

CA_SetHSMPolicy
Structure/Union member

CA_SetKCV
Structure/Union member

CA_SetLKCV
Structure/Union member

CA_SetMofN
Structure/Union member

CA_SetPedId
Structure/Union member

CA_SetRDK
Structure/Union member

CA_SetTPV
Structure/Union member

CA_SetTokenCertificateSignature
Structure/Union member

CA_SetTokenPolicies
Structure/Union member

CA_SetUserContainerName
Structure/Union member

CA_SpRawRead
Structure/Union member

CA_SpRawWrite
Structure/Union member

CA_SwitchSecondarySlot
Structure/Union member

CA_TimeSync
Structure/Union member

CA_TokenDelete
Structure/Union member

CA_TokenInsert
Structure/Union member

CA_TokenInsertNoAuth
Structure/Union member

CA_TokenZeroize
Structure/Union member

CA_UnloadModule
Structure/Union member

CA_UnlockClusteredSlot
Structure/Union member

CA_WaitForSlotEvent
Structure/Union member

CA_WriteCommonStore
Structure/Union member

CA_Zeroize
Structure/Union member

version
Structure/Union member

pycryptoki.cryptoki.**CK_SFNT_CA_FUNCTION_LIST_PTR**
alias of pycryptoki.cryptoki._ck_func_list.LP_CK_SFNT_CA_FUNCTION_LIST

pycryptoki.cryptoki.**CK_SFNT_CA_FUNCTION_LIST_PTR_PTR**
alias of pycryptoki.cryptoki._ck_func_list.LP_LP_CK_SFNT_CA_FUNCTION_LIST

class pycryptoki.cryptoki.**CK_SKIPJACK_PRIVATE_WRAP_PARAMS**

pBaseG
Structure/Union member

pPassword
Structure/Union member

pPrimeP
Structure/Union member

pPublicData
Structure/Union member

pRandomA
Structure/Union member

pSubprimeQ
Structure/Union member

ulPAndGLen
Structure/Union member

ulPublicDataLen
Structure/Union member

ulQLen
Structure/Union member

ulRandomLen
Structure/Union member

usPasswordLen
Structure/Union member

pycryptoki.cryptoki.**CK_SKIPJACK_PRIVATE_WRAP_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_SKIPJACK_PRIVATE_WRAP_PARAMS

class pycryptoki.cryptoki.**CK_SKIPJACK_RELAYX_PARAMS**

pNewPassword
Structure/Union member

pNewPublicData
Structure/Union member

pNewRandomA
Structure/Union member

pOldPassword
Structure/Union member

pOldPublicData
Structure/Union member

pOldRandomA
Structure/Union member

pOldWrappedX
Structure/Union member

ulNewPasswordLen
Structure/Union member

ulNewPublicDataLen
Structure/Union member

ulNewRandomLen
Structure/Union member

ulOldPasswordLen
Structure/Union member

ulOldPublicDataLen
Structure/Union member

ulOldRandomLen
Structure/Union member

ulOldWrappedXLen
Structure/Union member

pycryptoki.cryptoki.**CK_SKIPJACK_RELAYX_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_SKIPJACK_RELAYX_PARAMS

pycryptoki.cryptoki.**CK_SLOT_ID**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_SLOT_ID_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

class pycryptoki.cryptoki.**CK_SLOT_INFO**

firmwareVersion
Structure/Union member

flags
Structure/Union member

hardwareVersion
Structure/Union member

manufacturerID
Structure/Union member

slotDescription
Structure/Union member

pycryptoki.cryptoki.**CK_SLOT_INFO_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_SLOT_INFO

class pycryptoki.cryptoki.**CK_SSL3_KEY_MAT_OUT**

hClientKey
Structure/Union member

hClientMacSecret
Structure/Union member

hServerKey
Structure/Union member

hServerMacSecret
Structure/Union member

pIVClient
Structure/Union member

pIVServer
Structure/Union member

pycryptoki.cryptoki.**CK_SSL3_KEY_MAT_OUT_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_SSL3_KEY_MAT_OUT

class pycryptoki.cryptoki.**CK_SSL3_KEY_MAT_PARAMS**

RandomInfo
Structure/Union member

bIsExport
Structure/Union member

pReturnedKeyMaterial
Structure/Union member

ulIVSizeInBits
Structure/Union member

ulKeySizeInBits
Structure/Union member

ulMacSizeInBits
Structure/Union member

pycryptoki.cryptoki.**CK_SSL3_KEY_MAT_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_SSL3_KEY_MAT_PARAMS

class pycryptoki.cryptoki.**CK_SSL3_MASTER_KEY_DERIVE_PARAMS**

RandomInfo
Structure/Union member

pVersion
Structure/Union member

pycryptoki.cryptoki.**CK_SSL3_MASTER_KEY_DERIVE_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_SSL3_MASTER_KEY_DERIVE_PARAMS

class pycryptoki.cryptoki.**CK_SSL3_RANDOM_DATA**

pClientRandom
Structure/Union member

pServerRandom
Structure/Union member

ulClientRandomLen
Structure/Union member

ulServerRandomLen
Structure/Union member

pycryptoki.cryptoki.**CK_STATE**
alias of ctypes.c_ulong

class pycryptoki.cryptoki.**CK_TLS_PRF_PARAMS**

pLabel
Structure/Union member

pOutput
Structure/Union member

pSeed
Structure/Union member

pulOutputLen
Structure/Union member

ulLabelLen
Structure/Union member

ulSeedLen
Structure/Union member

pycryptoki.cryptoki.**CK_TLS_PRF_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_TLS_PRF_PARAMS

class pycryptoki.cryptoki.**CK_TOKEN_INFO**

firmwareVersion
Structure/Union member

flags
Structure/Union member

hardwareVersion
Structure/Union member

label
Structure/Union member

manufacturerID
Structure/Union member

model
Structure/Union member

serialNumber
Structure/Union member

ulFreePrivateMemory
Structure/Union member

ulFreePublicMemory
Structure/Union member

ulTotalPrivateMemory
Structure/Union member

ulTotalPublicMemory
Structure/Union member

usMaxPinLen
Structure/Union member

usMaxRwSessionCount
Structure/Union member

usMaxSessionCount
Structure/Union member

usMinPinLen
Structure/Union member

usRwSessionCount
Structure/Union member

usSessionCount
Structure/Union member

utcTime
Structure/Union member

pycryptoki.cryptoki.**CK_TOKEN_INFO_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_TOKEN_INFO

pycryptoki.cryptoki.**CK_ULONG**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_ULONG_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

pycryptoki.cryptoki.**CK_UNLOCKMUTEX**
alias of ctypes.CFUNCTYPE.<locals>.CFunctionType

pycryptoki.cryptoki.**CK_USER_TYPE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_USHORT**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_USHORT_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

pycryptoki.cryptoki.**CK_UTF8CHAR**
alias of ctypes.c_ubyte

pycryptoki.cryptoki.**CK_UTF8CHAR_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ubyte

class pycryptoki.cryptoki.**CK_VERSION**

major
Structure/Union member

minor
Structure/Union member

pycryptoki.cryptoki.**CK_VERSION_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_VERSION

pycryptoki.cryptoki.**CK_VOID_PTR**
alias of ctypes.c_void_p

pycryptoki.cryptoki.**CK_VOID_PTR_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_void_p

class pycryptoki.cryptoki.**CK_WTLS_KEY_MAT_OUT**

hKey
Structure/Union member

hMacSecret
Structure/Union member

pIV
Structure/Union member

pycryptoki.cryptoki.**CK_WTLS_KEY_MAT_OUT_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_WTLS_KEY_MAT_OUT

class pycryptoki.cryptoki.**CK_WTLS_KEY_MAT_PARAMS**

DigestMechanism
Structure/Union member

RandomInfo
Structure/Union member

bIsExport
Structure/Union member

pReturnedKeyMaterial
Structure/Union member

ulIVSizeInBits
Structure/Union member

ulKeySizeInBits
Structure/Union member

ulMacSizeInBits
Structure/Union member

ulSequenceNumber
Structure/Union member

pycryptoki.cryptoki.**CK_WTLS_KEY_MAT_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_WTLS_KEY_MAT_PARAMS

class pycryptoki.cryptoki.**CK_WTLS_MASTER_KEY_DERIVE_PARAMS**

DigestMechanism
Structure/Union member

RandomInfo
Structure/Union member

pVersion
Structure/Union member

pycryptoki.cryptoki.**CK_WTLS_MASTER_KEY_DERIVE_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_WTLS_MASTER_KEY_DERIVE_PARAMS

class pycryptoki.cryptoki.**CK_WTLS_PRF_PARAMS**

DigestMechanism
Structure/Union member

pLabel
Structure/Union member

pOutput
Structure/Union member

pSeed
Structure/Union member

pulOutputLen
Structure/Union member

ulLabelLen
Structure/Union member

ulSeedLen
Structure/Union member

pycryptoki.cryptoki.**CK_WTLS_PRF_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_WTLS_PRF_PARAMS

class pycryptoki.cryptoki.**CK_WTLS_RANDOM_DATA**

pClientRandom
Structure/Union member

pServerRandom
Structure/Union member

ulClientRandomLen
Structure/Union member

ulServerRandomLen
Structure/Union member

pycryptoki.cryptoki.**CK_WTLS_RANDOM_DATA_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_WTLS_RANDOM_DATA

class pycryptoki.cryptoki.**CK_X9_42_DH1_DERIVE_PARAMS**

kdf
Structure/Union member

pOtherInfo
Structure/Union member

pPublicData
Structure/Union member

ulOtherInfoLen
Structure/Union member

ulPublicDataLen
Structure/Union member

pycryptoki.cryptoki.**CK_X9_42_DH1_DERIVE_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_X9_42_DH1_DERIVE_PARAMS

class pycryptoki.cryptoki.**CK_X9_42_DH2_DERIVE_PARAMS**

hPrivateData
Structure/Union member

kdf
Structure/Union member

pOtherInfo
Structure/Union member

pPublicData
Structure/Union member

pPublicData2
Structure/Union member

ulOtherInfoLen
Structure/Union member

ulPrivateDataLen
Structure/Union member

ulPublicDataLen
Structure/Union member

ulPublicDataLen2
Structure/Union member

pycryptoki.cryptoki.**CK_X9_42_DH2_DERIVE_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_X9_42_DH2_DERIVE_PARAMS

pycryptoki.cryptoki.**CK_X9_42_DH_KDF_TYPE**
alias of ctypes.c_ulong

pycryptoki.cryptoki.**CK_X9_42_DH_KDF_TYPE_PTR**
alias of pycryptoki.cryptoki.c_defs.LP_c_ulong

class pycryptoki.cryptoki.**CK_X9_42_MQV_DERIVE_PARAMS**

hPrivateData
Structure/Union member

kdf
Structure/Union member

pOtherInfo
Structure/Union member

pPublicData
Structure/Union member

pPublicData2
Structure/Union member

publicKey
Structure/Union member

ulOtherInfoLen
Structure/Union member

ulPrivateDataLen
Structure/Union member

ulPublicDataLen
Structure/Union member

ulPublicDataLen2
Structure/Union member

pycryptoki.cryptoki.**CK_X9_42_MQV_DERIVE_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_X9_42_MQV_DERIVE_PARAMS

class pycryptoki.cryptoki.**CK_XOR_BASE_DATA_KDF_PARAMS**

kdf
Structure/Union member

pSharedData
Structure/Union member

ulSharedDataLen
Structure/Union member

pycryptoki.cryptoki.**CK_XOR_BASE_DATA_KDF_PARAMS_PTR**
alias of pycryptoki.cryptoki.ck_defs.LP_CK_XOR_BASE_DATA_KDF_PARAMS

pycryptoki.cryptoki.**C_CancelFunction**(**args*)
Cryptoki DLL call to C_CancelFunction.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_CloseAllSessions**(**args*)
Cryptoki DLL call to C_CloseAllSessions.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_CloseSession**(**args*)
Cryptoki DLL call to C_CloseSession.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_CopyObject**(**args*)
Cryptoki DLL call to C_CopyObject.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_CK_ATTRIBUTE
- **arg4** – c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_CreateObject**(**args*)
Cryptoki DLL call to C_CreateObject.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_ATTRIBUTE
- **arg3** – c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_Decrypt**(**args*)
Cryptoki DLL call to C_Decrypt.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DecryptDigestUpdate**(**args*)
Cryptoki DLL call to C_DecryptDigestUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DecryptFinal**(**args*)
Cryptoki DLL call to C_DecryptFinal.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DecryptInit**(**args*)
Cryptoki DLL call to C_DecryptInit.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DecryptUpdate**(**args*)
Cryptoki DLL call to C_DecryptUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DecryptVerifyUpdate**(**args*)
Cryptoki DLL call to C_DecryptVerifyUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DeriveKey**(**args*)
Cryptoki DLL call to C_DeriveKey.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong
- **arg4** – LP_CK_ATTRIBUTE
- **arg5** – c_ulong

- **arg6** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DestroyObject**(**args*)
Cryptoki DLL call to C_DestroyObject.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_Digest**(**args*)
Cryptoki DLL call to C_Digest.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DigestEncryptUpdate**(**args*)
Cryptoki DLL call to C_DigestEncryptUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DigestFinal**(**args*)
Cryptoki DLL call to C_DigestFinal.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DigestInit**(**args*)
Cryptoki DLL call to C_DigestInit.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM

Returns c_ulong

pycryptoki.cryptoki.**C_DigestKey**(**args*)
Cryptoki DLL call to C_DigestKey.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_DigestUpdate**(**args*)
Cryptoki DLL call to C_DigestUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_Encrypt**(**args*)
Cryptoki DLL call to C_Encrypt.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_EncryptFinal**(**args*)
Cryptoki DLL call to C_EncryptFinal.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_EncryptInit**(**args*)
Cryptoki DLL call to C_EncryptInit.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_EncryptUpdate**(**args*)
Cryptoki DLL call to C_EncryptUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_Finalize**(**args*)
Cryptoki DLL call to C_Finalize.

Parameters **arg1** – c_void_p

Returns c_ulong

pycryptoki.cryptoki.**C_FindObjects**(**args*)
Cryptoki DLL call to C_FindObjects.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – c_ulong
- **arg4** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_FindObjectsFinal**(**args*)
Cryptoki DLL call to C_FindObjectsFinal.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_FindObjectsInit**(**args*)
Cryptoki DLL call to C_FindObjectsInit.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_ATTRIBUTE
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_GenerateKey**(**args*)
Cryptoki DLL call to C_GenerateKey.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – LP_CK_ATTRIBUTE

- **arg4** – c_ulong
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_GenerateKeyPair**(**args*)
Cryptoki DLL call to C_GenerateKeyPair.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – LP_CK_ATTRIBUTE
- **arg4** – c_ulong
- **arg5** – LP_CK_ATTRIBUTE
- **arg6** – c_ulong
- **arg7** – LP_c_ulong
- **arg8** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_GenerateRandom**(**args*)
Cryptoki DLL call to C_GenerateRandom.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_GetAttributeValue**(**args*)
Cryptoki DLL call to C_GetAttributeValue.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_CK_ATTRIBUTE
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_GetFunctionList**(**args*)
Cryptoki DLL call to C_GetFunctionList.

Parameters **arg1** – LP_LP_CK_FUNCTION_LIST

Returns c_ulong

pycryptoki.cryptoki.**C_GetFunctionStatus**(**args*)
Cryptoki DLL call to C_GetFunctionStatus.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_GetInfo**(**args*)
Cryptoki DLL call to C_GetInfo.

Parameters **arg1** – LP_CK_INFO

Returns c_ulong

pycryptoki.cryptoki.**C_GetMechanismInfo**(**args*)
Cryptoki DLL call to C_GetMechanismInfo.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_CK_MECHANISM_INFO

Returns c_ulong

pycryptoki.cryptoki.**C_GetMechanismList**(**args*)
Cryptoki DLL call to C_GetMechanismList.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_GetObjectSize**(**args*)
Cryptoki DLL call to C_GetObjectSize.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_GetOperationState**(**args*)
Cryptoki DLL call to C_GetOperationState.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_GetSessionInfo**(**args*)
Cryptoki DLL call to C_GetSessionInfo.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_SESSION_INFO

Returns c_ulong

pycryptoki.cryptoki.**C_GetSlotInfo**(**args*)
Cryptoki DLL call to C_GetSlotInfo.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_SLOT_INFO

Returns c_ulong

pycryptoki.cryptoki.**C_GetSlotList**(**args*)
Cryptoki DLL call to C_GetSlotList.

Parameters

- **arg1** – c_ubyte
- **arg2** – LP_c_ulong
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_GetTokenInfo**(**args*)
Cryptoki DLL call to C_GetTokenInfo.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_TOKEN_INFO

Returns c_ulong

pycryptoki.cryptoki.**C_InitPIN**(**args*)
Cryptoki DLL call to C_InitPIN.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_InitToken**(**args*)
Cryptoki DLL call to C_InitToken.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte

Returns c_ulong

pycryptoki.cryptoki.**C_Initialize**(**args*)
Cryptoki DLL call to C_Initialize.

Parameters **arg1** – c_void_p

Returns c_ulong

pycryptoki.cryptoki.**C_Login**(**args*)
Cryptoki DLL call to C_Login.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_c_ubyte
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_Logout**(**args*)
Cryptoki DLL call to C_Logout.

Parameters **arg1** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_OpenSession**(**args*)
Cryptoki DLL call to C_OpenSession.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – c_void_p
- **arg4** – CFunctionType
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_SeedRandom**(**args*)
Cryptoki DLL call to C_SeedRandom.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_SetAttributeValue**(**args*)
Cryptoki DLL call to C_SetAttributeValue.

Parameters

- **arg1** – c_ulong
- **arg2** – c_ulong
- **arg3** – LP_CK_ATTRIBUTE
- **arg4** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_SetOperationState**(**args*)
Cryptoki DLL call to C_SetOperationState.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_SetPIN**(**args*)
Cryptoki DLL call to C_SetPIN.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_Sign**(**args*)
Cryptoki DLL call to C_Sign.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_SignEncryptUpdate**(**args*)
Cryptoki DLL call to C_SignEncryptUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_SignFinal**(**args*)
Cryptoki DLL call to C_SignFinal.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_SignInit**(**args*)
Cryptoki DLL call to C_SignInit.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_SignRecover**(**args*)
Cryptoki DLL call to C_SignRecover.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_SignRecoverInit**(**args*)
Cryptoki DLL call to C_SignRecoverInit.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_SignUpdate**(**args*)
Cryptoki DLL call to C_SignUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_UnwrapKey**(**args*)
Cryptoki DLL call to C_UnwrapKey.

Parameters

- **arg1** – c_ulong

- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong
- **arg6** – LP_CK_ATTRIBUTE
- **arg7** – c_ulong
- **arg8** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_Verify**(**args*)
Cryptoki DLL call to C_Verify.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong
- **arg4** – LP_c_ubyte
- **arg5** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_VerifyFinal**(**args*)
Cryptoki DLL call to C_VerifyFinal.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_VerifyInit**(**args*)
Cryptoki DLL call to C_VerifyInit.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_VerifyRecover**(**args*)
Cryptoki DLL call to C_VerifyRecover.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

- **arg4** – LP_c_ubyte
- **arg5** – LP_c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_VerifyRecoverInit**(**args*)
Cryptoki DLL call to C_VerifyRecoverInit.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_VerifyUpdate**(**args*)
Cryptoki DLL call to C_VerifyUpdate.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ubyte
- **arg3** – c_ulong

Returns c_ulong

pycryptoki.cryptoki.**C_WaitForSlotEvent**(**args*)
Cryptoki DLL call to C_WaitForSlotEvent.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_c_ulong
- **arg3** – c_void_p

Returns c_ulong

pycryptoki.cryptoki.**C_WrapKey**(**args*)
Cryptoki DLL call to C_WrapKey.

Parameters

- **arg1** – c_ulong
- **arg2** – LP_CK_MECHANISM
- **arg3** – c_ulong
- **arg4** – c_ulong
- **arg5** – LP_c_ubyte
- **arg6** – LP_c_ulong

Returns c_ulong

class pycryptoki.cryptoki.**HSM_STATS_PARAMS**

ulHighValue
Structure/Union member

ulId
Structure/Union member

ulLowValue
Structure/Union member

1.4.11 Pycryptoki Daemon Package

Start `pycryptoki.daemon.rpyc_pycryptoki.py` on your remote client, then connect to it using *RemotePycryptokiClient*. You can then use the RemotePycryptokiClient as if it were local:

```
pycryptoki = RemotePycryptokiClient('10.2.96.130', port=8001)
pycryptoki.c_initialize_ex()  # Executed on the daemon!
session = pycryptoki.c_open_session_ex(0)
#etc
```

1.4.11.1 daemon.rpyc_pycryptoki

RPYC-based daemon that allows for remote execution of pycryptoki commands.

Start via `./rpyc_pycryptoki.py -i <ip> -p <port>` or `python rpyc_pycryptoki.py -i <ip> -p <port>`

All methods starting are useable via `rpyc_conn.root.<method>`

All methods ending with `_ex` will automatically check the return code from cryptoki & raise an exception if it is not CKR_OK. It will *NOT* give you the return code, instead just returning the second part of the regular return tuple:

```
c_open_session()     # Returns: (ret_code, session_handle)
c_open_session_ex()  # Returns: session_handle, raises exception if ret_code != CKR_OK
```

class pycryptoki.daemon.rpyc_pycryptoki.**PycryptokiService**
Bases: `rpyc.core.service.SlaveService`

This is the core service to expose over RPYC.

If you're working with pointers, you'll need to create the pointer in a function here rather than passing in a pointer from the client (pointers getting pickled makes no sense).

static c_close_all_sessions(*slot*)
Closes all the sessions on a given slot

Parameters **slot** – The slot to close all sessions on

Returns retcode

Return type int

static c_close_all_sessions_ex(*slot*)
Executes *c_close_all_sessions()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_close_session(*h_session*)
Closes a session

Parameters **h_session** (*int*) – Session handle

Returns retcode

Return type int

static c_close_session_ex(*h_session*)
Executes *c_close_session()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_copy_object(*h_session*, *h_object*, *template=None*)
Method to call the C_CopyObject cryptoki command.

Parameters

- **h_session** (*int*) – Session handle
- **h_object** (*int*) – Handle to the object to be cloned
- **template** (*dict*) – Template for the new object. Defaults to None

Returns (retcode, Handle to the new cloned object)

Return type tuple

static c_copy_object_ex(*h_session*, *h_object*, *template=None*)
Executes *c_copy_object()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_create_object(*h_session*, *template*)
Creates an object based on a given python template

Parameters

- **h_session** (*int*) – Session handle
- **template** (*dict*) – The python template which the object will be based on

Returns (retcode, the handle of the object)

Return type tuple

static c_create_object_ex(*h_session*, *template*)
Executes *c_create_object()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_decrypt(*h_session*, *h_key*, *encrypted_data*, *mechanism*, *output_buffer=None*)
Decrypt given data with the given key and mechanism.

Note: If data is a list or tuple of strings, multi-part decryption will be used.

Parameters

- **h_session** (*int*) – The session to use
- **h_key** (*int*) – The handle of the key to use to decrypt
- **encrypted_data** (*bytes*) – Data to be decrypted

Note: Data will be converted to hexadecimal by calling:

```
to_hex(from_bytestring(data))
```

If you need to pass in raw hex data, call:

```
to_bytestring(from_hex(hex-data))
```

References:

- *to_hex()*
- *from_hex()*
- *to_bytestring()*
- *from_bytestring()*

- **mechanism** – See the `parse_mechanism()` function for possible values.
- **output_buffer** (*list* / *int*) – Integer or list of integers that specify a size of output buffer to use for an operation. By default will query with NULL pointer buffer to get required size of buffer.

Returns (Retcode, Python bytestring of decrypted data))

Return type tuple

static c_decrypt_ex(*h_session*, *h_key*, *encrypted_data*, *mechanism*, *output_buffer=None*)

Executes *c_decrypt()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_derive_key(*h_session*, *h_base_key*, *template*, *mechanism=None*)

Derives a key from another key.

Parameters

- **h_session** (*int*) – Session handle
- **h_base_key** (*int*) – The base key
- **template** (*dict*) – A python template of attributes to set on derived key
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns The result code, The derived key's handle

static c_derive_key_ex(*h_session*, *h_base_key*, *template*, *mechanism=None*)
Executes *c_derive_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_destroy_object(*h_session*, *h_object_value*)
Deletes the object corresponsing to the passed in object handle

Parameters

- **h_session** (*int*) – Session handle
- **h_object_value** (*int*) – The handle of the object to delete

Returns Return code

static c_destroy_object_ex(*h_session*, *h_object_value*)
Executes *c_destroy_object()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_digest(*h_session*, *data_to_digest*, *digest_flavor*, *mechanism=None*, *output_buffer=None*)
Digests some data

Parameters

- **h_session** (*int*) – Session handle
- **data_to_digest** (*bytes*) – The data to digest, either a string or a list of strings. If this is a list a multipart operation will be used
- **digest_flavor** (*int*) – The flavour of the mechanism to digest (MD2, SHA-1, HAS-160, SHA224, SHA256, SHA384, SHA512)

- **mechanism** – See the `parse_mechanism()` function for possible values. If None will use digest flavor.
- **output_buffer** (*list* | *int*) – Integer or list of integers that specify a size of output buffer to use for an operation. By default will query with NULL pointer buffer to get required size of buffer.

Returns (retcode, a python string of the digested data)

Return type tuple

static c_digest_ex(*h_session*, *data_to_digest*, *digest_flavor*, *mechanism=None*, *output_buffer=None*)

Executes *c_digest()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_digest_key(*h_session*, *h_key*, *digest_flavor*, *mechanism=None*)

Digest a key

Parameters

- **h_session** (*int*) – Session handle
- **h_key** (*int*) – Key to digest
- **digest_flavor** (*int*) – Digest flavor
- **mechanism** – See the `parse_mechanism()` function for possible values. If None will use digest flavor.

static c_digest_key_ex(*h_session*, *h_key*, *digest_flavor*, *mechanism=None*)

Executes `c_digestkey()`, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_encrypt(*h_session*, *h_key*, *data*, *mechanism*, *output_buffer=None*)
Encrypts data with a given key and encryption flavor encryption flavors

---

Note: If data is a list or tuple of strings, multi-part encryption will be used.

Parameters

- **h_session** (*int*) – Current session
- **h_key** (*int*) – The key handle to encrypt the data with
- **data** – The data to encrypt, either a bytestring or a list of bytestrings. If this is a list a multipart operation will be used

  ---

  Note: This will be converted to hexadecimal by calling:

  ```
  to_hex(from_bytestring(data))
  ```

  If you need to pass in raw hex data, call:

  ```
  to_bytestring(from_hex(hex-data))
  ```

  References:

  - *to_hex()*
  - *from_hex()*
  - *to_bytestring()*
  - *from_bytestring()*

- **mechanism** – See the `parse_mechanism()` function for possible values.
- **output_buffer** (*list|int*) – Integer or list of integers that specify a size of output buffer to use for an operation. By default will query with NULL pointer buffer to get required size of buffer.

Returns (Retcode, Python bytestring of encrypted data)

Return type tuple

static c_encrypt_ex(*h_session*, *h_key*, *data*, *mechanism*, *output_buffer=None*)
Executes *c_encrypt()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_finalize()
Finalizes PKCS11 library.

Returns Cryptoki return code

static c_finalize_ex()
Executes *c_finalize()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_find_objects(*h_session*, *template*, *num_entries*)
Calls c_find_objects and c_find_objects_init to get a python dictionary of the objects found.

Parameters

- **h_session** (*int*) – Session handle
- **template** – A python dictionary of the object template to look for
- **num_entries** – The max number of entries to return

Returns Returns a list of handles of objects found

static c_find_objects_ex(*h_session*, *template*, *num_entries*)
Executes *c_find_objects()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_generate_key(*h_session*, *mechanism=None*, *template=None*)
Generates a symmetric key of a given flavor given the correct template.

Parameters

- **h_session** (*int*) – Session handle
- **template** (*dict*) – The template to use to generate the key
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns (retcode, generated key handle)

Rtype tuple

static c_generate_key_ex(*h_session*, *mechanism=None*, *template=None*)
Executes *c_generate_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_generate_key_pair(*h_session*, *mechanism=None*, *pbkey_template=None*, *prkey_template=None*)
Generates a private and public key pair for a given flavor, and given public and private key templates. The return value will be the handle for the key.

Parameters

- **h_session** (*int*) – Session handle
- **pbkey_template** (*dict*) – The public key template to use for key generation
- **prkey_template** (*dict*) – The private key template to use for key generation
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns (retcode, public key handle, private key handle)

Return type tuple

static c_generate_key_pair_ex(*h_session*, *mechanism=None*, *pbkey_template=None*, *prkey_template=None*)
Executes *c_generate_key_pair()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_generate_random(*h_session*, *length*)
Generates a sequence of random numbers

Parameters

- **h_session** (*int*) – Session handle
- **length** (*int*) – The length in bytes of the random number sequence

Returns (retcode, A string of random data)

Return type tuple

static c_generate_random_ex(*h_session*, *length*)
Executes *c_generate_random()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_get_attribute_value(*h_session*, *h_object*, *template*)
Calls C_GetAttrributeValue to get an attribute value based on a python template

Parameters

- **h_session** (*int*) – Session handle
- **h_object** – The handle of the object to get attributes for
- **template** – A python dictionary representing the template of the attributes to be retrieved

Returns A python dictionary representing the attributes returned from the HSM/library

static c_get_attribute_value_ex(*h_session*, *h_object*, *template*)
Executes *c_get_attribute_value()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_get_info()

Get general information about the Cryptoki Library

Returns a dictionary containing the following keys:

- cryptokiVersion
- manufacturerID
- flags
- libraryDescription
- libraryVersion

`cryptokiVersion` and `libraryVersion` are *CK_VERSION* structs, and the major/minor values can be accessed directly (`info['cryptokiVersion'].major == 2`)

Returns (retcode, info dictionary)

static c_get_info_ex()

Executes *c_get_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_get_mechanism_info(*slot*, *mechanism_type*)

Gets a mechanism's info

Parameters

- **slot** – The slot to query
- **mechanism_type** – The type of the mechanism to get the information for

Returns The result code, The mechanism info

static c_get_mechanism_info_ex(*slot*, *mechanism_type*)
Executes *c_get_mechanism_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_get_mechanism_list(*slot*)
Gets the list of mechanisms from the HSM

Parameters **slot** – The slot number to get the mechanism list on

Returns The result code, A python dictionary representing the mechanism list

static c_get_mechanism_list_ex(*slot*)
Executes *c_get_mechanism_list()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_get_ped_id(*slot*)
Get the PED ID for the given slot.

Parameters **slot** – slot number

Returns The result code and ID

static c_get_ped_id_ex(*slot*)
Executes *c_get_ped_id()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_get_session_info(*session*)
Get information about the given session.

Parameters **session** (*int*) – session handle

Returns (retcode, dictionary of session information)

Return type tuple

static c_get_session_info_ex(*session*)
Executes *c_get_session_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_get_slot_info(*slot*)
Get information about the given slot number.

Parameters **slot** (*int*) – Target slot

Returns Dictionary of slot information

static c_get_slot_info_ex(*slot*)
Executes *c_get_slot_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_get_slot_list(*token_present=True*)
Get a list of all slots.

Parameters **token_present** (*bool*) – If true, will only return slots that have a token present.

Returns List of slots

static c_get_slot_list_ex(*token_present=True*)
Executes *c_get_slot_list()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_get_token_info(*slot_id*, *rstrip=True*)
Gets the token info for a given slot id

Parameters

- **slot_id** (*int*) – Token slot ID
- **rstrip** (*bool*) – If true, will strip trailing whitespace from char data.

Returns (retcode, A python dictionary representing the token info)

Return type tuple

static c_get_token_info_ex(*slot_id*, *rstrip=True*)
Executes *c_get_token_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_init_pin(*h_session*, *pin*)
Initializes the PIN

Parameters

- **h_session** (*int*) – Session handle
- **pin** – pin to c_initialize

Returns THe result code

static c_init_pin_ex(*h_session*, *pin*)
Executes *c_init_pin()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_init_token(*slot_num*, *password*, *token_label='Main Token'*)
Initializes at token at a given slot with the proper password and label

Parameters

- **slot_num** – The index of the slot to c_initialize a token in
- **password** – The password to c_initialize the slot with
- **token_label** – The label to c_initialize the slot with (Default value = 'Main Token')

Returns The result code

static c_init_token_ex(*slot_num*, *password*, *token_label='Main Token'*)
Executes *c_init_token()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_initialize(*flags=None*, *init_struct=None*)
Initializes current process for use with PKCS11.

Some sample flags:

CKF_LIBRARY_CANT_CREATE_OS_THREADS CKF_OS_LOCKING_OK

See the PKCS11 documentation for more details.

Parameters

- **flags** (*int*) – Flags to be set within InitArgs Struct. (Default = None)
- **init_struct** – InitArgs structure (Default = None)

Returns Cryptoki return code.

static c_initialize_ex(*flags=None*, *init_struct=None*)
Executes *c_initialize()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_logout(*h_session*)
Logs out of a given session

Parameters **h_session** (*int*) – Session handle

Returns retcode

Return type int

static c_logout_ex(*h_session*)
Executes *c_logout()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_open_session(*slot_num*, *flags=6*)
Opens a session on the given slot

Parameters

- **slot_num** (*int*) – The slot to get a session on
- **flags** (*int*) – The flags to open the session with (Default value = (CKF_SERIAL_SESSION | CKF_RW_SESSION)

Returns (retcode, session handle)

Return type tuple

static c_open_session_ex(*slot_num*, *flags=6*)
Executes *c_open_session()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_performselftest(*slot*, *test_type*, *input_data*, *input_data_len*)
Test: Performs a self test for specified test type on a given slot.

Parameters

- **slot** – slot number
- **test_type** – type of test CK_ULONG
- **input_data** – pointer to input data CK_BYTE_PTR
- **input_data_len** – input data length CK_ULONG

Returns

the result code

[CK_SLOT_ID, CK_ULONG, CK_BYTE_PTR, CK_ULONG, CK_BYTE_PTR, CK_ULONG_PTR]

static c_performselftest_ex(*slot*, *test_type*, *input_data*, *input_data_len*)
Executes *c_performselftest()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_seed_random(*h_session*, *seed*)
Seeds the random number generator

Parameters

- **h_session** (*int*) – Session handle
- **seed** (*bytes*) – A python string of some seed

Returns retcode

Return type int

static c_seed_random_ex(*h_session*, *seed*)
Executes *c_seed_random()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_set_attribute_value(*h_session*, *h_object*, *template*)
Calls C_SetAttributeValue to set an attribute value based on a python template

Parameters

- **h_session** (*int*) – Session handle
- **h_object** – The handle of the object to get attributes for
- **template** – A python dictionary representing the template of the attributes to be written

Returns A python dictionary representing the attributes returned from the HSM/library

static c_set_attribute_value_ex(*h_session*, *h_object*, *template*)
Executes *c_set_attribute_value()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_set_ped_id(*slot*, *id*)
Set the PED ID for the given slot.

Parameters

- **slot** – slot number
- **id** – PED ID to use

Returns The result code

static c_set_ped_id_ex(*slot*, *id*)
Executes *c_set_ped_id()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_set_pin(*h_session*, *old_pass*, *new_pass*)
Allows a user to change their PIN

Parameters

- **h_session** (*int*) – Session handle
- **old_pass** – The user's old password
- **new_pass** – The user's desired new password

Returns The result code

static c_set_pin_ex(*h_session*, *old_pass*, *new_pass*)
Executes *c_set_pin()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_sign(*h_session*, *h_key*, *data_to_sign*, *mechanism*, *output_buffer=None*)
Signs the given data with given key and mechanism.

Note: If data is a list or tuple of strings, multi-part operations will be used.

Parameters

- **h_session** (*int*) – Session handle
- **data_to_sign** – The data to sign, either a string or a list of strings. If this is a list a multipart operation will be used (using C_...Update and C_...Final)

  ex:

  - "This is a proper argument of some data to use in the function"
  - ["This is another format of data this", "function will accept.", "It will operate on these strings in parts"]
- **h_key** (*int*) – The signing key
- **mechanism** – See the `parse_mechanism()` function for possible values.
- **output_buffer** (*list | int*) – Integer or list of integers that specify a size of output buffer to use for an operation. By default will query with NULL pointer buffer to get required size of buffer.

Returns (retcode, python string of signed data)

Return type tuple

static c_sign_ex(*h_session*, *h_key*, *data_to_sign*, *mechanism*, *output_buffer=None*)
Executes *c_sign()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_unwrap_key(*h_session*, *h_unwrapping_key*, *wrapped_key*, *key_template*, *mechanism*)
Unwrap a key from an encrypted data blob.

Parameters

- **h_session** (*int*) – The session to use
- **h_unwrapping_key** (*int*) – The wrapping key handle
- **wrapped_key** (*bytes*) – The wrapped key

Note: Data will be converted to hexadecimal by calling:

```
to_hex(from_bytestring(data))
```

If you need to pass in raw hex data, call:

```
to_bytestring(from_hex(hex-data))
```

References:

- *to_hex()*
- *from_hex()*
- *to_bytestring()*
- *from_bytestring()*

- **key_template** (*dict*) – The python template representing the new key's template
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns (Retcode, unwrapped key handle)

Return type tuple

static c_unwrap_key_ex(*h_session*, *h_unwrapping_key*, *wrapped_key*, *key_template*, *mechanism*)
Executes *c_unwrap_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static c_verify(*h_session*, *h_key*, *data_to_verify*, *signature*, *mechanism*)
Verifies data with the given signature, key and mechanism.

Note: If data is a list or tuple of strings, multi-part operations will be used.

Parameters

- **h_session** (*int*) – Session handle
- **data_to_verify** – The data to sign, either a string or a list of strings. If this is a list a multipart operation will be used (using C_...Update and C_...Final)

  ex:

  – "This is a proper argument of some data to use in the function"

  – ["This is another format of data this", "function will accept.", "It will operate on these strings in parts"]
- **signature** (*bytes*) – Signature with which to verify the data.
- **h_key** (*int*) – The verifying key
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns retcode of verify operation

static c_verify_ex(*h_session*, *h_key*, *data_to_verify*, *signature*, *mechanism*)

Executes *c_verify()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

static c_wrap_key(*h_session*, *h_wrapping_key*, *h_key*, *mechanism*, *output_buffer=None*)

Wrap a key off the HSM into an encrypted data blob.

Parameters

- **h_session** (*int*) – The session to use
- **h_wrapping_key** (*int*) – The handle of the key to use to wrap another key
- **h_key** (*int*) – The key to wrap based on the encryption flavor
- **mechanism** – See the `parse_mechanism()` function for possible values.

Returns (Retcode, python bytestring representing wrapped key)

Return type tuple

static c_wrap_key_ex(*h_session*, *h_wrapping_key*, *h_key*, *mechanism*, *output_buffer=None*)

Executes *c_wrap_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_assign_key(*h_session*, *h_object*)

Crypto Officer assigns a key

Parameters

- **h_session** – session handle
- **object** – key handle to assign

Returns Ret code

static ca_assign_key_ex(*h_session*, *h_object*)

Executes *ca_assign_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_authorize_key(*h_session*, *h_object*, *auth_data*)

User authorizes key within session or access for use

Parameters

- **h_session** – session handle
- **object** – key handle to authorize
- **auth_data** – authorization byte list, e.g. [11, 12, 13, ..]

Returns Ret code

static ca_authorize_key_ex(*h_session*, *h_object*, *auth_data*)

Executes *ca_authorize_key()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_clonemofn(*h_session*)

Clones MofN secret from one token to another.

Parameters **h_session** (*int*) – Session handle

Returns the result code

static ca_clonemofn_ex(*h_session*)

Executes *ca_clonemofn()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_close_application_id_v2(*slot*, *appid*)

Close the AccessID associated with the given slot.

Parameters

- **slot** – Slot #.
- **appid** – bytestring of length 16.

Returns Retcode.

static ca_close_application_id_v2_ex(*slot*, *appid*)

Executes *ca_close_application_id_v2()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_close_secure_token(*h_session*, *h_ID*)

Parameters

- **h_session** (*int*) – Session handle
- **h_ID** –

static ca_close_secure_token_ex(*h_session*, *h_ID*)

Executes *ca_close_secure_token()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_closeapplicationID(*slot*, *id_high*, *id_low*)

Close a given AppID on a slot.

Parameters

- **slot** (*int*) – Slot on which to close the APP ID
- **id_high** (*int*) – High value of App ID
- **id_low** (*int*) – Low value of App ID

Returns retcode

Return type int

static ca_closeapplicationID_ex(*slot*, *id_high*, *id_low*)

Executes *ca_closeapplicationID()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_create_container(*h_session*, *storage_size*, *password=None*, *label='Inserted Token'*)

Inserts a token into a slot without a Security Officer on the token

Parameters

- **h_session** (*int*) – Session handle
- **storage_size** – The storage size of the token (0 for undefined/unlimited)
- **password** – The password associated with the token (Default value = 'userpin')
- **label** – The label associated with the token (Default value = 'Inserted Token')

Returns The result code, The container number

static ca_create_container_ex(*h_session*, *storage_size*, *password=None*, *label='Inserted Token'*)

Executes *ca_create_container()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_createloginchallenge(*h_session*, *user_type*, *challenge*)

Creates a login challenge for the given user.

Parameters

- **h_session** (*int*) – Session handle
- **user_type** – user type
- **challenge** – challenge

Returns the result code

static ca_createloginchallenge_ex(*h_session*, *user_type*, *challenge*)

Executes *ca_createloginchallenge()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_delete_container_with_handle(*h_session*, *h_container*)

Delete a container by handle

Parameters

- **h_session** (*int*) – Session handle
- **h_container** – target container handle

Returns result code

static ca_delete_container_with_handle_ex(*h_session*, *h_container*)

Executes *ca_delete_container_with_handle()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_deleteremotepedvector(*h_session*)

Deletes a remote PED vector

Parameters **h_session** (*int*) – Session handle

Returns the result code

static ca_deleteremotepedvector_ex(*h_session*)

Executes *ca_deleteremotepedvector()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_derive_key_and_wrap(*h_session*, *derive_mechanism*, *h_base_key*, *derive_template*, *wrapping_key*, *wrap_mechanism*, *output_buffer=2048*)

Derive a key from the base key and wrap it off the HSM using the wrapping key

Parameters

- **h_session** (*int*) – The session to use
- **h_base_key** (*int*) – The base key
- **derive_template** (*dict*) – A python template of attributes to set on derived key
- **derive_mechanism** – See the `parse_mechanism()` function for possible values.
- **wrapping_key** (*int*) – The wrapping key based on the encryption flavor
- **wrap_mechanism** – See the `parse_mechanism()` function for possible values.
- **output_buffer** – The size of the wrapped key, defaulted to a cert size

Returns (Retcode, python bytestring representing wrapped key)

Return type tuple

static ca_derive_key_and_wrap_ex(*h_session*, *derive_mechanism*, *h_base_key*, *derive_template*, *wrapping_key*, *wrap_mechanism*, *output_buffer=2048*)

Executes *ca_derive_key_and_wrap()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_destroy_multiple_objects(*h_session*, *objects*)

Delete multiple objects corresponding to given object handles

Parameters

- **h_session** (*int*) – Session handle

- **objects** (*list*) – The handles of the objects to delete

Returns Return code

static ca_destroy_multiple_objects_ex(*h_session*, *objects*)

Executes *ca_destroy_multiple_objects()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_duplicatemofn(*h_session*)

Duplicates a set of M of N vectors.

Parameters **h_session** (*int*) – Session handle

Returns the result code

static ca_duplicatemofn_ex(*h_session*)

Executes *ca_duplicatemofn()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_extract(*h_session*, *mechanism*)

Parameters

- **h_session** (*int*) – Session handle
- **mechanism** – See the `parse_mechanism()` function for possible values.

static ca_extract_ex(*h_session*, *mechanism*)

Executes *ca_extract()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_factory_reset(*slot*)

Does a factory reset on a given slot

Parameters **slot** – The slot to do a factory reset on

Returns The result code

static ca_factory_reset_ex(*slot*)

Executes *ca_factory_reset()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_generatemofn(*h_session*, *m_value*, *vector_value*, *vector_count*, *is_secure_port_used*)

Generates MofN secret information on a token.

Parameters

- **h_session** (*int*) – Session handle
- **m_value** – m
- **vector_count** – number of vectors
- **is_secure_port_used** – is secure port used
- **vector_value** –

Returns the result code

static ca_generatemofn_ex(*h_session*, *m_value*, *vector_value*, *vector_count*, *is_secure_port_used*)

Executes *ca_generatemofn()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

static ca_get_application_id()
Get the current process's AccessID.

Returns retcode, bytestring tuple.

static ca_get_application_id_ex()
Executes *ca_get_application_id()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

static ca_get_container_capability_set(*slot*, *h_container*)
Get the container capabilities of the given slot.

Parameters

- **slot** (*int*) – target slot number
- **h_container** (*int*) – target container handle

Returns result code, {id: val} dict of capabilities (None if command failed)

static ca_get_container_capability_set_ex(*slot*, *h_container*)
Executes *ca_get_container_capability_set()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_container_capability_setting(*slot*, *h_container*, *capability_id*)
Get the value of a container's single capability

Parameters

- **slot** – slot ID of slot to query
- **h_container** – target container handle
- **capability_id** – capability ID

Returns result code, CK_ULONG representing capability active or not

static ca_get_container_capability_setting_ex(*slot*, *h_container*, *capability_id*)
Executes *ca_get_container_capability_setting()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_container_list(*slot*, *group_handle=0*, *container_type=0*)
Get list of containers.

Parameters

- **slot** – slot ID of the slot to query
- **group_handle** – group ID
- **container_type** – type of container

Returns result code, list of container handles

static ca_get_container_list_ex(*slot*, *group_handle=0*, *container_type=0*)
Executes *ca_get_container_list()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_container_name(*slot*, *h_container*)
Get a container's name

Parameters

- **slot** – target slot
- **h_container** – target container handle

Returns result code, container name string

static ca_get_container_name_ex(*slot*, *h_container*)
Executes *ca_get_container_name()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_container_policy_set(*slot*, *h_container*)
Get the policies of the given slot and container.

Parameters

- **slot** (*int*) – target slot number
- **h_container** (*int*) – target container handle

Returns result code, {id: val} dict of policies (None if command failed)

static ca_get_container_policy_set_ex(*slot*, *h_container*)
Executes *ca_get_container_policy_set()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_container_policy_setting(*slot*, *h_container*, *policy_id*)
Get the value of a container's single policy

Parameters

- **slot** – slot ID of slot to query
- **h_container** – target container handle
- **policy_id** – policy ID

Returns result code, CK_ULONG representing policy active or not

static ca_get_container_policy_setting_ex(*slot*, *h_container*, *policy_id*)
Executes *ca_get_container_policy_setting()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_container_status(*slot*, *h_container*)
Get a container's Status

Parameters

- **slot** – target slot
- **h_container** – target container handle

Returns result code, dict of flags, dict of failed logins

static ca_get_container_status_ex(*slot*, *h_container*)
Executes *ca_get_container_status()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_container_storage_information(*slot*, *h_container*)
Get a container's storage information

Parameters

- **slot** – target slot
- **h_container** – target container handle

Returns result code, dict of storage values

static ca_get_container_storage_information_ex(*slot*, *h_container*)
Executes *ca_get_container_storage_information()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_cv_firmware_version(*slot_id*)
Cryptovisor specific ca extension function to get cv fw version

Parameters **slot_id** – slot id

Returns tuple of return code and cv fw version

static ca_get_cv_firmware_version_ex(*slot_id*)
Executes *ca_get_cv_firmware_version()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_hsm_capability_set(*slot*)
Get the capabilities of the given slot.

Parameters **slot** (*int*) – Target slot number

Returns retcode, {id: val} dict of capabilities (None if command failed)

static ca_get_hsm_capability_set_ex(*slot*)
Executes *ca_get_hsm_capability_set()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_hsm_capability_setting(*slot*, *capability_id*)
Get the value of a single capability

Parameters

- **slot** – slot ID of slot to query
- **capability_id** – capability ID

Returns result code, CK_ULONG representing capability active or not

static ca_get_hsm_capability_setting_ex(*slot*, *capability_id*)
Executes *ca_get_hsm_capability_setting()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_hsm_policy_set(*slot*)
Get the policies of the given slot.

Parameters **slot** (*int*) – Target slot number

Returns retcode, {id: val} dict of policies (None if command failed)

static ca_get_hsm_policy_set_ex(*slot*)
Executes *ca_get_hsm_policy_set()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_hsm_policy_setting(*slot*, *policy_id*)
Get the value of a single policy

Parameters

- **slot** – slot ID of slot to query
- **policy_id** – policy ID

Returns result code, CK_ULONG representing policy active or not

static ca_get_hsm_policy_setting_ex(*slot*, *policy_id*)
Executes *ca_get_hsm_policy_setting()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_object_handle(*slot*, *session*, *objectouid*)
Calls CA_GetObjectHandle to get the object handle from OUID

Parameters

- **slot** – partition slot number
- **session** – session id that was opened to run the function
- **objectouid** – OUID, a string of the hex value that maps to object handle

Returns a tuple containing the return code and the object handle mapping the given OUID

static ca_get_object_handle_ex(*slot*, *session*, *objectouid*)
Executes *ca_get_object_handle()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_session_info(*session*)
ca extension function that returns session information

Parameters **session** – session handle

Returns tuple of return code and session info dict

static ca_get_session_info_ex(*session*)
Executes *ca_get_session_info()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_time(*h_session*)

Parameters **h_session** (*int*) – Session handle

static ca_get_time_ex(*h_session*)
Executes *ca_get_time()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_get_token_policies(*slot*)
Get the policies of the given slot.

Parameters **slot** (*int*) – Target slot number

Returns retcode, {id: val} dict of policies (None if command failed)

static ca_get_token_policies_ex(*slot*)
Executes *ca_get_token_policies()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_hainit(*h_session*, *h_key*)
Creates a login key pair on the primary token.

Parameters

- **h_session** (*int*) – Session handle
- **h_key** – the login private key

Returns the result code

static ca_hainit_ex(*h_session*, *h_key*)
Executes *ca_hainit()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_increment_failed_auth_count(*h_session*, *h_object*)
This function is called by HA group when auth failure happens on a key to sync up status. Here its defined mostly for testing purposes :param h_session: session handle :param object: key handle to update :return: Ret code

static ca_increment_failed_auth_count_ex(*h_session*, *h_object*)
Executes *ca_increment_failed_auth_count()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_init_audit(*slot*, *audit_pin*, *audit_label*)

Parameters

- **slot** –
- **audit_pin** –
- **audit_label** –

static ca_init_audit_ex(*slot*, *audit_pin*, *audit_label*)
Executes *ca_init_audit()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_initializeremotepedvector(*h_session*)
Initializes a remote PED vector

Parameters **h_session** (*int*) – Session handle

Returns the result code

static ca_initializeremotepedvector_ex(*h_session*)
Executes *ca_initializeremotepedvector()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_insert(*h_session*, *mechanism*)

Parameters

- **h_session** (*int*) – Session handle
- **mechanism** – See the `parse_mechanism()` function for possible values.

static ca_insert_ex(*h_session*, *mechanism*)
Executes *ca_insert()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_modifyusagecount(*h_session*, *h_object*, *command_type*, *value*)
Modifies CKA_USAGE_COUNT attribute of the object.

Parameters

- **h_session** (*int*) – Session handle
- **h_object** – object
- **command_type** – command type
- **value** – value

Returns the result code

static ca_modifyusagecount_ex(*h_session*, *h_object*, *command_type*, *value*)
Executes *ca_modifyusagecount()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

static ca_mtkresplit(*slot*)
Resplit the MTK

Parameters **slot** – slot number

Returns the result code

static ca_mtkresplit_ex(*slot*)
Executes *ca_mtkresplit()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

static ca_mtkrestore(*slot*)
Restore the MTK

Parameters **slot** – slot number

Returns the result code

static ca_mtkrestore_ex(*slot*)
Executes *ca_mtkrestore()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_mtkzeroize(*slot*)
Zeroize the MTK

Parameters **slot** – slot number

Returns the result code

static ca_mtkzeroize_ex(*slot*)
Executes *ca_mtkzeroize()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_open_application_id_v2(*slot*, *appid*)
Open the given AccessID for the target slot.

Parameters

- **slot** – Slot #.
- **appid** – bytestring of length 16.

Returns Retcode.

static ca_open_application_id_v2_ex(*slot*, *appid*)
Executes *ca_open_application_id_v2()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_open_secure_token(*h_session*, *storage_path*, *dev_ID*, *mode*)

Parameters

- **h_session** (*int*) – Session handle
- **storage_path** –
- **dev_ID** –
- **mode** –

static ca_open_secure_token_ex(*h_session*, *storage_path*, *dev_ID*, *mode*)

Executes *ca_open_secure_token()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_openapplicationID(*slot*, *id_high*, *id_low*)

Open an application ID on the given slot.

Parameters

- **slot** (*int*) – Slot on which to open the APP ID
- **id_high** (*int*) – High value of App ID
- **id_low** (*int*) – Low value of App ID

Returns retcode

Return type int

static ca_openapplicationID_ex(*slot*, *id_high*, *id_low*)

Executes *ca_openapplicationID()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_read_all_utilization_counters(*h_session*)
Read Metrics from previously saved HSM snapshot Call either functions prior to create snapshot: ca_read_utilization_metrics ca_read_and_reset_utilization_metrics

Returns a dictionary, where keys are serial numbers

and values are dictionaries of bins and values, example: 'SIGN':0

static ca_read_all_utilization_counters_ex(*h_session*)
Executes *ca_read_all_utilization_counters()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_read_and_reset_utilization_metrics(*session*)
HSM reads current utilization data and saves as a snapshot; HSM resets metrics to zeroes

Parameters **session** – session id that was opened to run the function

Returns a dictionary with partition serial numbers as keys, value - dictionary of utilization metrics

static ca_read_and_reset_utilization_metrics_ex(*session*)
Executes *ca_read_and_reset_utilization_metrics()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_read_utilization_metrics(*session*)
HSM reads utilization data and saves as a snapshot

Parameters **session** – session id that was opened to run the function

Returns Ret code

static ca_read_utilization_metrics_ex(*session*)
Executes *ca_read_utilization_metrics()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_reset_authorization_data(*h_session*, *h_object*, *auth_data*)
CO resets auth data on unassigned key

Parameters

- **h_session** – session handle
- **object** – key handle to update
- **auth_data** – byte list, e.g. [11, 12, 13, ..]

Returns Ret code

static ca_reset_authorization_data_ex(*h_session*, *h_object*, *auth_data*)
Executes *ca_reset_authorization_data()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_restart(*slot*)

Parameters **slot** –

static ca_restart_ex(*slot*)
Executes *ca_restart()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_set_authorization_data(*h_session*, *h_object*, *old_auth_data*, *new_auth_data*)
User changes authorization data on key object (private, secret)

Parameters

- **h_session** – session handle
- **object** – key handle to update
- **old_auth_data** – byte list, e.g. [11, 12, 13, ..]
- **new_auth_data** – byte list, e.g. [11, 12, 13, ..]

Returns Ret code

static ca_set_authorization_data_ex(*h_session*, *h_object*, *old_auth_data*, *new_auth_data*)
Executes *ca_set_authorization_data()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_set_container_policies(*h_session*, *h_container*, *policies*)
Set multiple container policies.

Parameters

- **h_session** (*int*) – Session handle
- **h_container** – target container handle
- **policies** – dict of policy ID ints and value ints

Returns result code

static ca_set_container_policies_ex(*h_session*, *h_container*, *policies*)
Executes *ca_set_container_policies()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

static ca_set_container_policy(*h_session*, *h_containerber*, *policy_id*, *policy_val*)
Sets a policy on the container.

NOTE: With per partition SO this method should generally not be used. Instead ca_set_partition_policies should be used

Parameters

- **h_session** (*int*) – Session handle
- **h_containerber** – The container number to set the policy on.
- **policy_id** – The identifier of the policy (ex. CONTAINER_CONFIG_MINIMUM_PIN_LENGTH)
- **policy_val** – The value to set the policy to

Returns The result code

static ca_set_container_policy_ex(*h_session*, *h_containerber*, *policy_id*, *policy_val*)
Executes *ca_set_container_policy()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

static ca_set_container_size(*h_session*, *h_container*, *size*)
Set a container's size

Parameters

- **h_session** (*int*) – Session handle
- **h_container** – target container handle
- **size** – size

Returns result code

static ca_set_container_size_ex(*h_session*, *h_container*, *size*)
Executes *ca_set_container_size()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_set_destructive_hsm_policies(*h_session*, *policies*)
Set multiple HSM policies.

Parameters

- **h_session** (*int*) – Session handle
- **policies** – dict of policy ID ints and value ints

Returns result code

static ca_set_destructive_hsm_policies_ex(*h_session*, *policies*)
Executes *ca_set_destructive_hsm_policies()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_set_destructive_hsm_policy(*h_session*, *policy_id*, *policy_val*)
Sets the destructive HSM policies by calling CA_SetDestructiveHSMPolicy

Parameters

- **h_session** (*int*) – Session handle
- **policy_id** – The ID of the policy being set
- **policy_val** – The value of the policy being set

Returns The result code

static ca_set_destructive_hsm_policy_ex(*h_session*, *policy_id*, *policy_val*)
Executes *ca_set_destructive_hsm_policy()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_set_hsm_policies(*h_session*, *policies*)
Set multiple HSM policies.

Parameters

- **h_session** (*int*) – Session handle
- **policies** – dict of policy ID ints and value ints

Returns result code

static ca_set_hsm_policies_ex(*h_session*, *policies*)
Executes *ca_set_hsm_policies()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_set_hsm_policy(*h_session*, *policy_id*, *policy_val*)
Sets the HSM policies by calling CA_SetHSMPolicy

Parameters

- **h_session** (*int*) – Session handle
- **policy_id** – The ID of the policy being set
- **policy_val** – The value of the policy being set

Returns The result code

static ca_set_hsm_policy_ex(*h_session*, *policy_id*, *policy_val*)
Executes *ca_set_hsm_policy()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_setapplicationID(*id_high*, *id_low*)
Set the App ID for the current process.

Parameters

- **id_high** (*int*) – High value of App ID
- **id_low** (*int*) – Low value of App ID

Returns retcode

Return type int

static ca_setapplicationID_ex(*id_high*, *id_low*)
Executes *ca_setapplicationID()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_settokencertificatesignature(*h_session*, *access_level*, *customer_id*, *pub_template*, *signature*, *signature_len*)
Completes the installation of a certificate on a token. The caller must supply a public key and a signature for token certificate. The public key is provided through the template; it must contain a key type, a modulus and a public exponent.

Parameters

- **h_session** (*int*) – Session handle
- **access_level** – the access level

- **customer_id** – the customer ID
- **pub_template** – the public template
- **signature** – the signature
- **signature_len** – the length in bytes of the signature

Returns the result code

static ca_settokencertificatesignature_ex(*h_session*, *access_level*, *customer_id*, *pub_template*, *signature*, *signature_len*)

Executes *ca_settokencertificatesignature()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_sim_extract(*h_session*, *key_handles*, *authform*, *auth_secrets=None*, *subset_size=0*, *delete_after_extract=False*)

Extract multiple keys to a wrapped blob. The returned blob can then be written into a file.

Parameters

- **h_session** (*int*) – Session handle
- **key_handles** (*list[int]*) – List of key handles to extract
- **authform** (*int*) – Type of authentication to use. See `pycryptoki.backup.SIM_AUTH` for details
- **auth_secrets** (*list(str)*) – Authorization secrets to use (Length will correspond to the `N` value in ckdemo)
- **subset_size** (*int*) – Subset size required for key use (Corresponds to the `M` value in ckdemo)
- **delete_after_extract** (*bool*) – If true, will destroy the original keys after they have been extracted.

Returns retcode, blob_data tuple.

static ca_sim_extract_ex(*h_session*, *key_handles*, *authform*, *auth_secrets=None*, *subset_size=0*, *delete_after_extract=False*)

Executes *ca_sim_extract()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_sim_insert(*h_session*, *blob_data*, *authform*, *auth_secrets=None*)

Insert keys into the HSM from blob data that was wrapped off using SIM.

Parameters

- **h_session** (*int*) – Session handle
- **blob_data** (*str*) – Read in raw wrapped data. Typically read in from a file.
- **authform** (*int*) – Type of authentication to use. See `pycryptoki.backup.SIM_AUTH` for details
- **auth_secrets** (*list[str]*) – Authorization secrets to use (Length will correspond to the `N` value in ckdemo)

Returns retcode, keys tuple, where `keys` is a list of integers.

static ca_sim_insert_ex(*h_session*, *blob_data*, *authform*, *auth_secrets=None*)

Executes *ca_sim_insert()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_sim_multisign(*h_session*, *blob_data*, *data_to_sign*, *mechanism*, *authform*, *auth_secrets=None*)

Sign data using keys that were extracted to a SIM blob.

Parameters

- **h_session** (*int*) – Session handle
- **blob_data** (*str*) – Read in raw wrapped key data. Typically read in from a file.
- **data_to_sign** – List of bytestring data to sign
- **mechanism** – Mechanism to use with the Sign operation
- **authform** (*int*) – Type of authentication to use. See `pycryptoki.backup.SIM_AUTH` for details

- **auth_secrets** (*list[str]*) – Authorization secrets to use (Length will correspond to the N value in ckdemo)

Returns retcode, signature list

static ca_sim_multisign_ex(*h_session*, *blob_data*, *data_to_sign*, *mechanism*, *authform*, *auth_secrets=None*)

Executes *ca_sim_multisign()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static ca_time_sync(*h_session*, *ultime*)

Parameters

- **h_session** (*int*) – Session handle
- **ultime** –

static ca_time_sync_ex(*h_session*, *ultime*)

Executes *ca_time_sync()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

exposed_ca_authorize_key

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

class C: @staticmethod def f(arg1, arg2, ...):

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

exposed_ca_authorize_key_ex
staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

class C: @staticmethod def f(arg1, arg2, ...):

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

exposed_ca_set_authorization_data
staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

class C: @staticmethod def f(arg1, arg2, ...):

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

exposed_ca_set_authorization_data_ex
staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

class C: @staticmethod def f(arg1, arg2, ...):

...

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

static get_token_by_label(*label*)
Iterates through all the tokens and returns the first token that has a label that is identical to the one that is passed in

Parameters **label** – The label of the token to search for

Returns The result code, The slot of the token

static get_token_by_label_ex(*label*)
Executes *get_token_by_label()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static login(*h_session*, *slot_num=1*, *password=None*, *user_type=1*)
Login to the given session.

Parameters

- **h_session** (*int*) – Session handle
- **slot_num** (*int*) – Slot index to login on (Default value = 1)
- **password** (*bytes*) – Password to login with (Default value = "userpin")
- **user_type** (*int*) – User type to login as (Default value = 1)

Returns retcode

Return type int

static login_ex(*h_session*, *slot_num=1*, *password=None*, *user_type=1*)
Executes *login()*, and checks the retcode; raising an exception if the return code is not CKR_OK.

Note: By default, this will not return the return code if the function returns additional data.

Example:

```
retcode, key_handle = c_generate_key(...)
#vs
key_handle = c_generate_key_ex(...)
```

If the function *only* returns the retcode, then that will still be returned:

```
retcode = c_seed_random(...)
retcode = c_seed_random_ex(...)
```

---

static test_attrs(*attributes*)
Function used for validating that dicts can be used across rpyc pipes.

static test_conn()
Test Function used to validate that custom functions are properly exposed over RPYC. Specifically not using something like conn.ping() to verify exposed functions.

static to_bool(*val*, *reverse=False*)
Convert a boolean-ish value to a pValue, ulValueLen tuple.

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to *pycryptoki.cryptoki.CK_BBOOL*, ctypes.c_ulong size of bool value)

static to_byte_array(*val*, *reverse=False*)

Converts an arbitrarily sized integer, list, or byte array into a byte array.

It'll zero-pad the bit length so it's a multiple of 8, then convert the int to binary, split the binary string into sections of 8, then place each section into a slot in a ctypes.c_ubyte array (converting to small int).

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to *pycryptoki.cryptoki.CK_BYTE* array, ctypes.c_ulong size of array)

static to_char_array(*val*, *reverse=False*)

Convert the given string or list of string values into a char array.

This is slightly different than to_byte_array, which has different assumptions as to the format of the input.

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to *pycryptoki.cryptoki.CK_CHAR* array, ctypes.c_ulong size of array)

static to_ck_date(*val*, *reverse=False*)

Transform a date string, date dictionary, or date object into a PKCS11 readable form (YYYYMMDD)

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to *pycryptoki.cryptoki.CK_CHAR* array, ctypes.c_ulong size of array)

static to_long(*val*, *reverse=False*)

Convert a integer/long value to a pValue, ulValueLen tuple

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to ctypes.c_ulong, ctypes.c_ulong size of long value)

static to_subattributes(*val*, *reverse=False*)

Convert to another Attributes class & return the struct.

Parameters

- **val** – Value to convert
- **reverse** – Whether to convert from C -> Python

Returns (ctypes.c_void_p ptr to *pycryptoki.cryptoki.CK_ATTRIBUTE* array, ctypes.c_ulong size of array)

pycryptoki.daemon.rpyc_pycryptoki.**configure_logging**(*logfile=None*)
Setup logging. If a log file is specified, will log to that file.

Parameters **logfile** (*str*) – Log file path/name to use for logging.

Returns Configured logger.

pycryptoki.daemon.rpyc_pycryptoki.**create_server_subprocess**(*target*, *args*, *logger*)
Create the subprocess, set it as a daemon, setup a signal handler in case the parent process is killed, the child process should also be killed, then return the subprocess.

Parameters

- **target** – Target function to run in a subprocess
- **args** – Args to pass to the function

Returns *multiprocessing.Process*

pycryptoki.daemon.rpyc_pycryptoki.**server_launch**(*service*, *ip*, *port*, *config*)
Target for the multiprocessing Pycryptoki service.

Parameters

- **service** –
- **ip** –
- **port** –
- **config** –

Returns

1.4.11.2 pycryptoki.pycryptoki_client

class pycryptoki.pycryptoki_client.**LocalPycryptokiClient**
Bases: object

Class forwards calls to pycryptoki to local client but looks identical to remote client

cleanup()

kill()

class pycryptoki.pycryptoki_client.**RemotePycryptokiClient**(*ip=None*, *port=None*)
Bases: object

Class to handle connecting to a remote Pycryptoki RPYC daemon.

After instantiation, you can use it directly to make calls to a remote cryptoki library via RPYC (no need to do any imports or anything like that, just use the direct pycryptoki call like client.c_initialize_ex())

Parameters

- **ip** – IP Address of the client the remote daemon is running on.

- **port** – What Port the daemon is running on.

cleanup()

kill()
Close out the local RPYC connection.

start()
Start the connection to the remote RPYC daemon.

started
Check if the RPYC connection is alive.

Returns boolean

pycryptoki.pycryptoki_client.**connection_test**(*func*)
Decorator to check that the underlying rpyc connection is alive before sending commands across it.

Parameters **func** –

Returns

pycryptoki.pycryptoki_client.**log_args**(*funcname*, *arg_dict*)
This will run through each of the key, value pairs of the argument spec passed into pycryptoki and perform the following checks:

- if key is a template, format the template data through a dict lookup
- if key is password, set the log data to be '*'
- if value is longer than 40 characters, abbreviate it.

Parameters **arg_dict** –

Returns

pycryptoki.pycryptoki_client.**retry**(*ExceptionToCheck*, *tries=4*, *delay=3*, *backoff=2*, *logger=None*)
Retry calling the decorated function using an exponential backoff.

http://www.saltycrane.com/blog/2009/11/trying-out-retry-decorator-python/ original from: http://wiki.python.org/moin/PythonDecoratorLibrary#Retry

Parameters

- **ExceptionToCheck** (*Exception or tuple*) – the exception to check. may be a tuple of exceptions to check
- **tries** (*int*) – number of times to try (not retry) before giving up
- **delay** (*int*) – initial delay between retries in seconds
- **backoff** (*int*) – backoff multiplier e.g. value of 2 will double the delay each retry
- **logger** (*logging.Logger instance*) – logger to use. If None, print

Python Module Index

p

Index

D

E

F

G

H

I

K

L

M

N

P

R

S

T

U

V

Y