Plug-And-Trust-Nano-Package API Document

.

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 se05x_scp03.h File Reference

Se05x SCP03 utils.

```
#include <stdint.h>
```

Functions

- smStatus_t Se05x_API_SCP03_GetSessionKeys (pSe05xSession_t session_ctx, uint8_t ∗encKey, size_↩
t ∗encKey_len, uint8_t ∗macKey, size_t ∗macKey_len, uint8_t ∗rMacKey, size_t ∗rMacKey_len)
- smStatus_t Se05x_API_SCP03_GetMcvCounter (pSe05xSession_t pSessionCtx, uint8_t ∗pCounter, size↩
_t ∗pCounterLen, uint8_t ∗pMcv, size_t ∗pMcvLen)
- smStatus_t Se05x_API_SCP03_SetSessionKeys (pSe05xSession_t session_ctx, const uint8_t ∗encKey, const size_t encKey_len, const uint8_t ∗macKey, const size_t macKey_len, const uint8_t ∗rMacKey, const size_t rMacKey_len)
- smStatus_t Se05x_API_SCP03_SetMcvCounter (pSe05xSession_t pSessionCtx, const uint8_t ∗pCounter, const size_t counterLen, const uint8_t ∗pMcv, const size_t mcvLen)

2.1.1 Detailed Description

Se05x SCP03 utils.

Copyright 2021,2022 NXP SPDX-License-Identifier: Apache-2.0

2.1.2 Function Documentation

2.1.2.1 Se05x_API_SCP03_GetMcvCounter()

```
smStatus_t Se05x_API_SCP03_GetMcvCounter (
            pSe05xSession_t pSessionCtx,
            uint8_t ∗ pCounter,
            size_t ∗ pCounterLen,
            uint8_t ∗ pMcv,
            size_t ∗ pMcvLen )
```

Se05x_API_SCP03_GetMcvCounter

Get SCP03 MCV and Counter values.

Parameters

in	*session_ctx*	The session context
in,out	*pCounter*	SCP03 Counter
in,out	*pCounterLen*	SCP03 Counter length
in,out	*pMcv*	SCP03 MCV
in,out	*pMcvLen*	SCP03 MCV length

Returns

The sm status.

2.1.2.2 Se05x_API_SCP03_GetSessionKeys()

```
smStatus_t Se05x_API_SCP03_GetSessionKeys (
            pSe05xSession_t session_ctx,
            uint8_t * encKey,
            size_t * encKey_len,
            uint8_t * macKey,
            size_t * macKey_len,
            uint8_t * rMacKey,
            size_t * rMacKey_len )
```

Se05x_API_SCP03_GetSessionKeys

Get SCP03 session keys.

Parameters

in	*session_ctx*	The session context
in,out	*encKey*	Enc key buffer
in,out	*encKey_len*	Enc key buffer length
in,out	*macKey*	Mac key buffer
in,out	*macKey_len*	Mac key buffer length
in,out	*rMacKey*	Rmac key buffer
in,out	*rMacKey_len*	Rmac key buffer length

Returns

The sm status.

2.1.2.3 Se05x_API_SCP03_SetMcvCounter()

```
smStatus_t Se05x_API_SCP03_SetMcvCounter (
            pSe05xSession_t pSessionCtx,
            const uint8_t * pCounter,
            const size_t counterLen,
            const uint8_t * pMcv,
            const size_t mcvLen )
```

Se05x_API_SCP03_SetMcvCounter

Set SCP03 MCV and Counter values.

Parameters

in	*session_ctx*	The session context
in	*pCounter*	SCP03 Counter
in	*pCounterLen*	SCP03 Counter length
in	*pMcv*	SCP03 MCV
in	*pMcvLen*	SCP03 MCV length

Returns

The sm status.

2.1.2.4 Se05x_API_SCP03_SetSessionKeys()

```
smStatus_t Se05x_API_SCP03_SetSessionKeys (
            pSe05xSession_t session_ctx,
            const uint8_t * encKey,
            const size_t encKey_len,
            const uint8_t * macKey,
            const size_t macKey_len,
            const uint8_t * rMacKey,
            const size_t rMacKey_len )
```

Se05x_API_SCP03_SetSessionKeys

Set SCP03 session keys.

Parameters

in	*session_ctx*	The session context
in	*encKey*	Enc key buffer
in	*encKey_len*	Enc key buffer length
in	*macKey*	Mac key buffer
in	*macKey_len*	Mac key buffer length
in	*rMacKey*	Rmac key buffer
in	*rMacKey_len*	Rmac key buffer length

Returns

The sm status.

2.2 se05x_scp03.h

Go to the documentation of this file.

```c

#ifndef SE05X_SCP03_H_INC
#define SE05X_SCP03_H_INC

/* ********************** Include files ********************** */
#include <stdint.h>

/* ********************** Function Prototypes ********************** */

smStatus_t Se05x_API_SCP03_GetSessionKeys(pSe05xSession_t session_ctx,
    uint8_t *encKey,
    size_t *encKey_len,
    uint8_t *macKey,
    size_t *macKey_len,
    uint8_t *rMacKey,
    size_t *rMacKey_len);

smStatus_t Se05x_API_SCP03_GetMcvCounter(
    pSe05xSession_t pSessionCtx, uint8_t *pCounter, size_t *pCounterLen, uint8_t *pMcv, size_t *pMcvLen);

smStatus_t Se05x_API_SCP03_SetSessionKeys(pSe05xSession_t session_ctx,
    const uint8_t *encKey,
    const size_t encKey_len,
    const uint8_t *macKey,
    const size_t macKey_len,
    const uint8_t *rMacKey,
    const size_t rMacKey_len);

smStatus_t Se05x_API_SCP03_SetMcvCounter(pSe05xSession_t pSessionCtx,
    const uint8_t *pCounter,
    const size_t counterLen,
    const uint8_t *pMcv,
    const size_t mcvLen);

smStatus_t Se05x_API_SCP03_CalculateMacRspApdu(
    pSe05xSession_t session_ctx, uint8_t *inData, size_t inDataLen, uint8_t *outSignature, size_t
   *outSignatureLen);

smStatus_t Se05x_API_SCP03_CalculateMacCmdApdu(
    pSe05xSession_t session_ctx, uint8_t *inData, size_t inDataLen, uint8_t *outSignature, size_t
   *outSignatureLen);

smStatus_t Se05x_API_SCP03_PadCommandAPDU(pSe05xSession_t session_ctx, uint8_t *cmdBuf, size_t
   *pCmdBufLen);

smStatus_t Se05x_API_SCP03_CalculateCommandICV(pSe05xSession_t session_ctx, uint8_t *pIcv);

smStatus_t Se05x_API_SCP03_GetResponseICV(pSe05xSession_t session_ctx, uint8_t *pIcv, bool hasCmd);

smStatus_t Se05x_API_SCP03_RestoreSwRAPDU(pSe05xSession_t session_ctx,
    uint8_t *rspBuf,
    size_t *pRspBufLen,
    uint8_t *plaintextResponse,
    size_t plaintextRespLen,
    uint8_t *sw);

void Se05x_API_SCP03_IncCommandCounter(pSe05xSession_t session_ctx);
/* ********************** Constants ********************** */

#define INITIAL_HOST_CHALLANGE                         \
    {                                                  \
        0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88 \
    }

#define SCP_GP_IU_KEY_DIV_DATA_LEN 10
#define SCP_GP_IU_KEY_INFO_LEN 3
#define SCP_GP_CARD_CHALLENGE_LEN 8
#define SCP_GP_HOST_CHALLENGE_LEN 8
#define SCP_GP_IU_CARD_CRYPTOGRAM_LEN 8
#define SCP_GP_IU_SEQ_COUNTER_LEN 3
#define SCP_GP_SW_LEN 2
#define CRYPTO_KEY_CHECK_LEN (3)

#define ASN_ECC_NIST_256_HEADER_LEN 26
#define KEY_PARAMETER_REFERENCE_TAG 0xF0
#define KEY_PARAMETER_REFERENCE_VALUE_LEN 0x01 // Fixed for Nist256key
#define KEY_PARAMETER_REFERENCE_VALUE 0x03     // key parameter value need to check in the spec it is 00
#define GPCS_KEY_TYPE_ECC_NIST256 0xB0
#define GPCS_KEY_TYPE_AES 0x88
#define GPCS_KEY_LEN_AES 16

#define SCP_ID 0xAB
```

```
#define SCP_CONFIG 0x01

#define SCP_MCV_LEN 16 // MAC Chaining Length

#define CLA_ISO7816 (0x00)
#define CLA_GP_7816 (0x80)
#define CLA_GP_SECURITY_BIT (0x04)

#define INS_GP_INITIALIZE_UPDATE (0x50)
#define INS_GP_EXTERNAL_AUTHENTICATE (0x82)
#define INS_GP_SELECT (0xA4)
#define INS_GP_PUT_KEY (0xD8)
#define INS_GP_INTERNAL_AUTHENTICATE (0x88)
#define INS_GP_GET_DATA (0xCA)
#define P1_GP_GET_DATA (0xBF)
#define P2_GP_GET_DATA (0x21)

/* Sizes used in SCP */
#define AES_KEY_LEN_nBYTE (16)

#define SCP_KEY_SIZE (16)
#define SCP_CMAC_SIZE (16)       // length of the CMAC calculated (and used as MAC chaining value)
#define SCP_IV_SIZE (16)         // length of the Inital Vector
#define SCP_COMMAND_MAC_SIZE (8) // length of the MAC appended in the APDU payload (8 'MSB's)

#define DATA_CARD_CRYPTOGRAM (0x00)
#define DATA_HOST_CRYPTOGRAM (0x01)
#define DATA_DERIVATION_SENC (0x04)
#define DATA_DERIVATION_SMAC (0x06)
#define DATA_DERIVATION_SRMAC (0x07)
#define DATA_DERIVATION_INITIAL_MCV (0x08)
#define DATA_DERIVATION_L_64BIT (0x0040)
#define DATA_DERIVATION_L_128BIT (0x0080)
#define DATA_DERIVATION_KDF_CTR (0x01)

#define DD_LABEL_LEN 12

/* defines used to indicate the command type */
#define C_MAC (0x01)
#define C_ENC (0x02)
#define R_MAC (0x10)
#define R_ENC (0x20)

#define SECLVL_CDEC_RENC_CMAC_RMAC (0x33)

#define SCP_DATA_PAD_BYTE 0x80

#define CMAC_SIZE (8)

#define SCP_OK (SW_OK)
#define SCP_UNDEFINED_CHANNEL_ID (0x7041)
#define SCP_FAIL (0x7042)
#define SCP_CARD_CRYPTOGRAM_FAILS_TO_VERIFY (0x7043)
#define SCP_PARAMETER_ERROR (0x7044)

#define NO_C_MAC_NO_C_ENC_NO_R_MAC_NO_R_ENC 0
#define C_MAC_NO_C_ENC_R_MAC_NO_R_ENC (C_MAC | R_MAC)
#define C_MAC_C_ENC_R_MAC_R_ENC (C_MAC | C_ENC | R_MAC | R_ENC)
#define SECURITY_LEVEL C_MAC_C_ENC_R_MAC_R_ENC

#define APPLET_SCP_INIT_UPDATE_LEN 0x0D
#define APPLET_SCP_EXT_AUTH_LEN 0x15

#define CONTEXT_LENGTH (SCP_GP_HOST_CHALLENGE_LEN + SCP_GP_CARD_CHALLENGE_LEN)
#define DAA_BUFFER_LEN (CONTEXT_LENGTH + DD_LABEL_LEN + 16)
#endif //#ifndef SE05X_SCP03_H_INC
```

2.3 se05x_APDU_apis.h File Reference

Se05x apdu functions.

```
#include "se05x_types.h"
#include "se05x_tlv.h"
```

Functions

- smStatus_t Se05x_API_SessionOpen (pSe05xSession_t session_ctx)
- smStatus_t Se05x_API_SessionClose (pSe05xSession_t session_ctx)
- smStatus_t Se05x_API_WriteECKey (pSe05xSession_t session_ctx, pSe05xPolicy_t policy, SE05x_Max←Attemps_t maxAttempt, uint32_t objectID, SE05x_ECCurve_t curveID, const uint8_t ∗privKey, size_t priv←KeyLen, const uint8_t ∗pubKey, size_t pubKeyLen, const SE05x_INS_t ins_type, const SE05x_KeyPart_t key_part)
- smStatus_t Se05x_API_ReadObject (pSe05xSession_t session_ctx, uint32_t objectID, uint16_t offset, uint16_t length, uint8_t ∗data, size_t ∗pdataLen)
- smStatus_t Se05x_API_GetVersion (pSe05xSession_t session_ctx, uint8_t ∗pappletVersion, size_←t ∗appletVersionLen)
- smStatus_t Se05x_API_ECDSASign (pSe05xSession_t session_ctx, uint32_t objectID, SE05x_←ECSignatureAlgo_t ecSignAlgo, const uint8_t ∗inputData, size_t inputDataLen, uint8_t ∗signature, size_t ∗psignatureLen)
- smStatus_t Se05x_API_ECDSAVerify (pSe05xSession_t session_ctx, uint32_t objectID, SE05x_←ECSignatureAlgo_t ecSignAlgo, const uint8_t ∗inputData, size_t inputDataLen, const uint8_t ∗signature, size_t signatureLen, SE05x_Result_t ∗presult)
- smStatus_t Se05x_API_CheckObjectExists (pSe05xSession_t session_ctx, uint32_t objectID, SE05x_←Result_t ∗presult)
- smStatus_t Se05x_API_WriteBinary (pSe05xSession_t session_ctx, pSe05xPolicy_t policy, uint32_t object←ID, uint16_t offset, uint16_t length, const uint8_t ∗inputData, size_t inputDataLen)
- smStatus_t Se05x_API_ECDHGenerateSharedSecret (pSe05xSession_t session_ctx, uint32_t objectID, const uint8_t ∗pubKey, size_t pubKeyLen, uint8_t ∗sharedSecret, size_t ∗psharedSecretLen)
- smStatus_t Se05x_API_CipherOneShot (pSe05xSession_t session_ctx, uint32_t objectID, SE05x_Cipher←Mode_t cipherMode, const uint8_t ∗inputData, size_t inputDataLen, uint8_t ∗IV, size_t IVLen, uint8_←t ∗outputData, size_t ∗poutputDataLen, const SE05x_Cipher_Oper_OneShot_t operation)

  Se05x_API_CipherOneShot.
- smStatus_t Se05x_API_WriteSymmKey (pSe05xSession_t session_ctx, pSe05xPolicy_t policy, SE05x_←MaxAttemps_t maxAttempt, uint32_t objectID, SE05x_KeyID_t kekID, const uint8_t ∗keyValue, size_t key←ValueLen, const SE05x_INS_t ins_type, const SE05x_SymmKeyType_t type)
- smStatus_t Se05x_API_DeleteSecureObject (pSe05xSession_t session_ctx, uint32_t objectID)

2.3.1 Detailed Description

Se05x apdu functions.

Copyright 2021,2022 NXP SPDX-License-Identifier: Apache-2.0

2.3.2 Function Documentation

2.3.2.1 Se05x_API_CheckObjectExists()

```
smStatus_t Se05x_API_CheckObjectExists (
            pSe05xSession_t session_ctx,
            uint32_t objectID,
            SE05x_Result_t * presult )
```

Se05x_API_CheckObjectExists

Check if a Secure Object with a certain identifier exists or not.

Command to Applet

```
* +-------+------------+-----------------------------------------+
* | Field | Value      | Description                             |
* +=======+============+=========================================+
* | CLA   | 0x80       |                                         |
* +-------+------------+-----------------------------------------+
* | INS   | INS_MGMT   | See :cpp:type:`SE05x_INS_t`             |
* +-------+------------+-----------------------------------------+
* | P1    | P1_DEFAULT | See :cpp:type:`SE05x_P1_t`              |
* +-------+------------+-----------------------------------------+
* | P2    | P2_EXIST   | See :cpp:type:`SE05x_P2_t`              |
* +-------+------------+-----------------------------------------+
* | Lc    | #(Payload) |                                         |
* +-------+------------+-----------------------------------------+
* |       | TLV[TAG_1] | 4-byte existing Secure Object identifier. |
* +-------+------------+-----------------------------------------+
* | Le    | 0x00       |                                         |
* +-------+------------+-----------------------------------------+
*
```

R-APDU Body

```
* +------------+----------------------------------+
* | Value      | Description                      |
* +============+==================================+
* | TLV[TAG_1] | 1-byte :cpp:type:`SE05x_Result_t` |
* +------------+----------------------------------+
*
```

R-APDU Trailer

```
* +-------------+--------------------------------+
* | SW          | Description                    |
* +=============+================================+
* | SW_NO_ERROR | Data is returned successfully. |
* +-------------+--------------------------------+
*
```

Parameters

in	*session_ctx*	Session Context [0:kSE05x_pSession]
in	*objectID*	object id [1:kSE05x_TAG_1]
out	*presult*	[0:kSE05x_TAG_1]

2.3.2.2 Se05x_API_CipherOneShot()

```
smStatus_t Se05x_API_CipherOneShot (
            pSe05xSession_t session_ctx,
            uint32_t objectID,
            SE05x_CipherMode_t cipherMode,
            const uint8_t * inputData,
            size_t inputDataLen,
            uint8_t * IV,
            size_t IVLen,
            uint8_t * outputData,
            size_t * poutputDataLen,
            const SE05x_Cipher_Oper_OneShot_t operation )
```

Se05x_API_CipherOneShot.

Encrypt or decrypt data in one shot mode.

The key object must be either an AES key or DES key.

Command to Applet

```
* +---------+-----------------------+-----------------------------------------------+
* | Field   | Value                 | Description                                   |
* +=========+=======================+===============================================+
* | CLA     | 0x80                  |                                               |
* +---------+-----------------------+-----------------------------------------------+
* | INS     | INS_CRYPTO            | :cpp:type:`SE05x_INS_t`                       |
* +---------+-----------------------+-----------------------------------------------+
* | P1      | P1_CIPHER             | See :cpp:type:`SE05x_P1_t`                    |
* +---------+-----------------------+-----------------------------------------------+
* | P2      | P2_ENCRYPT_ONESHOT or | See :cpp:type:`SE05x_P2_t`                    |
* |         | P2_DECRYPT_ONESHOT    |                                               |
* +---------+-----------------------+-----------------------------------------------+
* | Lc      | #(Payload)            |                                               |
* +---------+-----------------------+-----------------------------------------------+
* | Payload | TLV[TAG_1]            | 4-byte identifier of the key object.          |
* +---------+-----------------------+-----------------------------------------------+
* |         | TLV[TAG_2]            | 1-byte CipherMode                             |
* +---------+-----------------------+-----------------------------------------------+
* |         | TLV[TAG_3]            | Byte array containing input data.             |
* +---------+-----------------------+-----------------------------------------------+
* |         | TLV[TAG_4]            | Byte array containing an initialization       |
* |         |                       | vector.   [Optional]   [Conditional: only when |
* |         |                       | the Crypto Object type equals CC_CIPHER and    |
* |         |                       | subtype is not including ECB]                 |
* +---------+-----------------------+-----------------------------------------------+
* | Le      | 0x00                  | Expecting return data.                        |
* +---------+-----------------------+-----------------------------------------------+
*
```

R-APDU Body

```
* +------------+-------------+
* | Value      | Description |
* +============+=============+
* | TLV[TAG_1] | Output data |
* +------------+-------------+
*
```

R-APDU Trailer

```
* +-------------+--------------------------------------+
* | SW          | Description                          |
* +=============+======================================+
* | SW_NO_ERROR | The command is handled successfully. |
* +-------------+--------------------------------------+
*
```

Parameters

in	*session_ctx*	The session context
in	*objectID*	The object id (AES key object with key length = 128 or 192 or 256 bits)

Parameters

in	*cipherMode*	The cipher mode
in	*inputData*	The input data (16 Bytes aligned data. Max - 112 Bytes)
in	*inputDataLen*	The input data length
in	*IV*	Initial vector (16 Bytes)
in	*IVLen*	The iv length
in,out	*outputData*	The output data
in,out	*poutputDataLen*	The poutput data length
in	*operation*	The operation

Returns

The sm status.

2.3.2.3 Se05x_API_DeleteSecureObject()

```
smStatus_t Se05x_API_DeleteSecureObject (
            pSe05xSession_t session_ctx,
            uint32_t objectID )
```

Se05x_API_DeleteSecureObject

Deletes a Secure Object.

If the object origin = ORIGIN_PROVISIONED, an error will be returned and the object is not deleted.

Command to Applet

```
* +-------+------------------+------------------------------------------+
* | Field | Value            | Description                              |
* +=======+==================+==========================================+
* | CLA   | 0x80             |                                          |
* +-------+------------------+------------------------------------------+
* | INS   | INS_MGMT         | See :cpp:type:`SE05x_INS_t`              |
* +-------+------------------+------------------------------------------+
* | P1    | P1_DEFAULT       | See :cpp:type:`SE05x_P1_t`               |
* +-------+------------------+------------------------------------------+
* | P2    | P2_DELETE_OBJECT | See :cpp:type:`SE05x_P2_t`               |
* +-------+------------------+------------------------------------------+
* | Lc    | #(Payload)       |                                          |
* +-------+------------------+------------------------------------------+
* |       | TLV[TAG_1]       | 4-byte existing Secure Object identifier. |
* +-------+------------------+------------------------------------------+
* | Le    | -                |                                          |
* +-------+------------------+------------------------------------------+
*
```

R-APDU Body

NA

R-APDU Trailer

```
* +-------------+----------------------------------------------+
* | SW          | Description                                  |
* +=============+==============================================+
* | SW_NO_ERROR | The file is created or updated successfully. |
* +-------------+----------------------------------------------+
*
```

Parameters

in	*session_ctx*	Session Context [0:kSE05x_pSession]
in	*objectID*	object id [1:kSE05x_TAG_1]

2.3.2.4 Se05x_API_ECDHGenerateSharedSecret()

```
smStatus_t Se05x_API_ECDHGenerateSharedSecret (
              pSe05xSession_t session_ctx,
              uint32_t objectID,
              const uint8_t * pubKey,
              size_t pubKeyLen,
              uint8_t * sharedSecret,
              size_t * psharedSecretLen )
```

Se05x_API_ECDHGenerateSharedSecret

The ECDHGenerateSharedSecret command generates a shared secret ECC point on the curve using an EC private key on SE05X and an external public key provided by the caller. The output shared secret is returned to the caller.

Command to Applet

```
* +------------+------------------------------+----------------------------------------------+
* | Field      | Value                        | Description                                  |
* +============+==============================+==============================================+
* | CLA        | 0x80                         |                                              |
* +------------+------------------------------+----------------------------------------------+
* | INS        | INS_CRYPTO                   | :cpp:type:`SE05x_INS_t`                      |
* +------------+------------------------------+----------------------------------------------+
* | P1         | P1_EC                        | See :cpp:type:`SE05x_P1_t`                   |
* +------------+------------------------------+----------------------------------------------+
* | P2         | P2_DH                        | See :cpp:type:`SE05x_P2_t`                   |
* +------------+------------------------------+----------------------------------------------+
* | Lc         | #(Payload)                   |                                              |
* +------------+------------------------------+----------------------------------------------+
* | Payload    | TLV[TAG_1]                   | 4-byte identifier of the key pair or private |
* |            |                              | key.                                         |
* +------------+------------------------------+----------------------------------------------+
* | TLV[TAG_2] | External public key (see     |                                              |
* |            | :cpp:type:`ECKeyRef`).       |                                              |
* +------------+------------------------------+----------------------------------------------+
* | TLV[TAG_7] | 4-byte HMACKey identifier to |                                              |
* |            | store output.    [Optional]  |                                              |
* +------------+------------------------------+----------------------------------------------+
* | Le         | 0x00                         | Expected shared secret length.               |
* +------------+------------------------------+----------------------------------------------+
*
```

R-APDU Body

```
* +------------+----------------------------------------------+
* | Value      | Description                                  |
* +============+==============================================+
* | TLV[TAG_1] | The returned shared secret.    [Conditional: |
* |            | only when the input does not contain         |
* |            | TLV[TAG_7].}                                 |
* +------------+----------------------------------------------+
*
```

R-APDU Trailer

```
* +-------------+--------------------------------------+
* | SW          | Description                          |
* +=============+======================================+
* | SW_NO_ERROR | The command is handled successfully. |
* +-------------+--------------------------------------+
*
```

Parameters

in	*session_ctx*	Session Context [0:kSE05x_pSession]
in	*objectID*	objectID [1:kSE05x_TAG_1]
in	*pubKey*	pubKey [2:kSE05x_TAG_2]
in	*pubKeyLen*	Length of pubKey
out	*sharedSecret*	[0:kSE05x_TAG_1]
in,out	*psharedSecretLen*	Length for sharedSecret

2.3.2.5 Se05x_API_ECDSASign()

```
smStatus_t Se05x_API_ECDSASign (
             pSe05xSession_t session_ctx,
             uint32_t objectID,
             SE05x_ECSignatureAlgo_t ecSignAlgo,
             const uint8_t * inputData,
             size_t inputDataLen,
             uint8_t * signature,
             size_t * psignatureLen )
```

Se05x_API_ECDSASign

The ECDSASign command signs external data using the indicated key pair or private key.

The ECSignatureAlgo indicates the ECDSA algorithm that is used, but the hashing of data always must be done on the host. E.g., if ECSignatureAlgo = SIG_ ECDSA_SHA256, the user must have applied SHA256 on the input data already. Supported SHA algorithms - SHA1, SHA224, SHA256, SHA384.

The user must take care of providing the correct input length; i.e., the data input length (TLV[TAG_3]) must match the digest indicated in the signature algorithm (TLV[TAG_2]).

In any case, the APDU payload must be smaller than MAX_APDU_PAYLOAD_LENGTH.

This is performed according to the ECDSA algorithm as specified in [ANSI X9.62]. The signature (a sequence of two integers 'r' and 's') as returned in the response adheres to the ASN.1 DER encoded formatting rules for integers.

Command to Applet

```
* +-------+--------------+------------------------------------------+
* | Field | Value        | Description                              |
* +=======+==============+==========================================+
* | CLA   | 0x80         |                                          |
* +-------+--------------+------------------------------------------+
* | INS   | INS_CRYPTO   | :cpp:type:`SE05x_INS_t`                  |
* +-------+--------------+------------------------------------------+
* | P1    | P1_SIGNATURE | See :cpp:type:`SE05x_P1_t`               |
* +-------+--------------+------------------------------------------+
* | P2    | P2_SIGN      | See :cpp:type:`SE05x_P2_t`               |
* +-------+--------------+------------------------------------------+
* | Lc    | #(Payload)   |                                          |
* +-------+--------------+------------------------------------------+
* |       | TLV[TAG_1]   | 4-byte identifier of EC key pair or private |
* |       |              | key.                                     |
* +-------+--------------+------------------------------------------+
* |       | TLV[TAG_2]   | 1-byte ECSignatureAlgo.                  |
* +-------+--------------+------------------------------------------+
* |       | TLV[TAG_3]   | Byte array containing input data.        |
* +-------+--------------+------------------------------------------+
* | Le    | 0x00         | Expecting ASN.1 signature                |
* +-------+--------------+------------------------------------------+
*
```

R-APDU Body

```
* +------------+---------------------------------+
* | Value      | Description                     |
* +============+=================================+
* | TLV[TAG_1] | ECDSA Signature in ASN.1 format. |
* +------------+---------------------------------+
*
```

R-APDU Trailer

```
* +-------------+-------------------------------------+
* | SW          | Description                         |
* +=============+=====================================+
* | SW_NO_ERROR | The command is handled successfully. |
* +-------------+-------------------------------------+
*
```

Parameters

in	*session_ctx*	Session Context [0:kSE05x_pSession]
in	*objectID*	objectID [1:kSE05x_TAG_1]
in	*ecSignAlgo*	ecSignAlgo [2:kSE05x_TAG_2]
in	*inputData*	inputData [3:kSE05x_TAG_3]
in	*inputDataLen*	Length of inputData
out	*signature*	[0:kSE05x_TAG_1]
in,out	*psignatureLen*	Length for signature

2.3.2.6 Se05x_API_ECDSAVerify()

```
smStatus_t Se05x_API_ECDSAVerify (
            pSe05xSession_t session_ctx,
            uint32_t objectID,
            SE05x_ECSignatureAlgo_t ecSignAlgo,
            const uint8_t * inputData,
            size_t inputDataLen,
            const uint8_t * signature,
            size_t signatureLen,
            SE05x_Result_t * presult )
```

Se05x_API_ECDSAVerify

The ECDSAVerify command verifies whether the signature is correct for a given (hashed) data input using an EC public key or EC key pair's public key.

The ECSignatureAlgo indicates the ECDSA algorithm that is used, but the hashing of data must always be done on the host. E.g., if ECSignatureAlgo = SIG_ ECDSA_SHA256, the user must have applied SHA256 on the input data already. Supported SHA algorithms - SHA1, SHA224, SHA256, SHA384.

The key cannot be passed externally to the command directly. In case users want to use the command to verify signatures using different public keys or the public key value regularly changes, the user should create a transient key object to which the key value is written and then the identifier of that transient secure object can be used by this ECDSAVerify command.

Command to Applet

```
* +-------+--------------+-----------------------------------------------+
* | Field | Value        | Description                                   |
* +=======+==============+===============================================+
* | CLA   | 0x80         |                                               |
* +-------+--------------+-----------------------------------------------+
* | INS   | INS_CRYPTO   | :cpp:type:`SE05x_INS_t`                       |
* +-------+--------------+-----------------------------------------------+
* | P1    | P1_SIGNATURE | See :cpp:type:`SE05x_P1_t`                    |
* +-------+--------------+-----------------------------------------------+
* | P2    | P2_VERIFY    | See :cpp:type:`SE05x_P2_t`                    |
* +-------+--------------+-----------------------------------------------+
* | Lc    | #(Payload)   |                                               |
* +-------+--------------+-----------------------------------------------+
* |       | TLV[TAG_1]   | 4-byte identifier of the key pair or public   |
* |       |              | key.                                          |
* +-------+--------------+-----------------------------------------------+
* |       | TLV[TAG_2]   | 1-byte ECSignatureAlgo.                       |
* +-------+--------------+-----------------------------------------------+
* |       | TLV[TAG_3]   | Byte array containing ASN.1 signature         |
* +-------+--------------+-----------------------------------------------+
* |       | TLV[TAG_5]   | Byte array containing hashed data to compare. |
* +-------+--------------+-----------------------------------------------+
* | Le    | 0x03         | Expecting TLV with :cpp:type:`SE05x_Result_t` |
* +-------+--------------+-----------------------------------------------+
*
```

R-APDU Body

```
* +------------+--------------------------------------+
* | Value      | Description                          |
* +============+======================================+
* | TLV[TAG_1] | Result of the signature verification |
* |            | (:cpp:type:`SE05x_Result_t`).        |
* +------------+--------------------------------------+
*
```

R-APDU Trailer

```
* +-----------------------------+--------------------------------------+
* | SW                          | Description                          |
* +=============================+======================================+
* | SW_NO_ERROR                 | The command is handled successfully. |
* +-----------------------------+--------------------------------------+
* | SW_CONDITIONS_NOT_SATISFIED | Incorrect data                       |
* +-----------------------------+--------------------------------------+
*
```

Parameters

in	*session_ctx*	Session Context [0:kSE05x_pSession]
in	*objectID*	objectID [1:kSE05x_TAG_1]
in	*ecSignAlgo*	ecSignAlgo [2:kSE05x_TAG_2]
in	*inputData*	inputData [3:kSE05x_TAG_3]
in	*inputDataLen*	Length of inputData
in	*signature*	signature [4:kSE05x_TAG_5]
in	*signatureLen*	Length of signature
out	*presult*	[0:kSE05x_TAG_1]

2.3.2.7 Se05x_API_GetVersion()

```
smStatus_t Se05x_API_GetVersion (
            pSe05xSession_t session_ctx,
            uint8_t * pappletVersion,
            size_t * appletVersionLen )
```

Se05x_API_GetVersion

Gets the applet version information.

This will return 7-byte VersionInfo (including major, minor and patch version of the applet, supported applet features and secure box version).

Command to Applet

```
* +-------+------------------------------+--------------------------------------------+
* | Field | Value                        | Description                                |
* +=======+==============================+============================================+
* | CLA   | 0x80                         |                                            |
* +-------+------------------------------+--------------------------------------------+
* | INS   | INS_MGMT                     | See :cpp:type:`SE05x_INS_t`                |
* +-------+------------------------------+--------------------------------------------+
* | P1    | P1_DEFAULT                   | See :cpp:type:`SE05x_P1_t`                 |
* +-------+------------------------------+--------------------------------------------+
* | P2    | P2_VERSION or P2_VERSION_EXT | See :cpp:type:`SE05x_P2_t`                 |
* +-------+------------------------------+--------------------------------------------+
* | Lc    | #(Payload)                   |                                            |
* +-------+------------------------------+--------------------------------------------+
* | Le    | 0x00                         | Expecting TLV with 7-byte data  (when P2 =  |
* |       |                              | P2_VERSION) or a TLV with 37 byte data (when |
* |       |                              | P2=  P2_VERSION_EXT).                      |
* +-------+------------------------------+--------------------------------------------+
*
```

R-APDU Body

```
* +------------+--------------------------------------------+
* | Value      | Description                                |
* +============+============================================+
* | TLV[TAG_1] | 7-byte :cpp:type:`VersionInfoRef` (if P2 =     |
* |            | P2_VERSION) or 7-byte  VersionInfo followed by |
* |            | 30 bytes extendedFeatureBits (if P2 =          |
* |            | P2_VERSION_EXT)                                |
* +------------+--------------------------------------------+
*
```

R-APDU Trailer

```
* +-------------+-------------------------------+
* | SW          | Description                   |
* +=============+===============================+
* | SW_NO_ERROR | Data is returned successfully. |
* +-------------+-------------------------------+
*
```

Parameters

in	*session_ctx*	The session context
	pappletVersion	The papplet version
	appletVersionLen	The applet version length

Returns

The sm status.

2.3.2.8 Se05x_API_ReadObject()

```
smStatus_t Se05x_API_ReadObject (
            pSe05xSession_t session_ctx,
            uint32_t objectID,
            uint16_t offset,
            uint16_t length,
            uint8_t * data,
            size_t * pdataLen )
```

Se05x_API_ReadObject

Reads the content of a Secure Object.

- If the object is a key pair, the command will return the key pair's public key.
- If the object is a public key, the command will return the public key.
- If the object is a private key or a symmetric key or a userID, the command will return SW_CONDITIONS_↩ NOT_SATISFIED.
- If the object is a binary file, the file content is read, giving the offset in TLV[TAG_2] and the length to read in TLV[TAG_3]. Both TLV[TAG_2] and TLV[TAG_3] are bound together; i.e.. either both tags are present, or both are absent. If both are absent, the whole file content is returned.

Command to Applet

```
* +-------+------------+---------------------------------------------+
* | Field | Value      | Description                                 |
* +=======+============+=============================================+
* | CLA   | 0x80       |                                             |
* +-------+------------+---------------------------------------------+
* | INS   | INS_READ   | See :cpp:type:`SE05x_INS_t`, in addition to  |
* |       |            | INS_READ, users can set the INS_ATTEST flag. |
* |       |            | In that case, attestation applies.          |
* +-------+------------+---------------------------------------------+
* | P1    | P1_DEFAULT | See :cpp:type:`SE05x_P1_t`                   |
* +-------+------------+---------------------------------------------+
* | P2    | P2_DEFAULT | See :cpp:type:`SE05x_P2_t`                   |
* +-------+------------+---------------------------------------------+
* | Lc    | #(Payload) | Payload Length.                             |
* +-------+------------+---------------------------------------------+
* |       | TLV[TAG_1] | 4-byte object identifier                    |
* +-------+------------+---------------------------------------------+
* |       | TLV[TAG_2] | 2-byte offset   [Optional: default 0]       |
* |       |            | [Conditional: only when the object is a     |
* |       |            | BinaryFile object]                          |
* +-------+------------+---------------------------------------------+
* |       | TLV[TAG_3] | 2-byte length   [Optional: default 0]       |
* |       |            | [Conditional: only when the object is a     |
* |       |            | BinaryFile object]                          |
* +-------+------------+---------------------------------------------+
* |       | TLV[TAG_4] | 1-byte :cpp:type:`SE05x_RSAKeyComponent_t`:  |
* |       |            | either RSA_COMP_MOD or RSA_COMP_PUB_EXP.    |
* |       |            | [Optional]   [Conditional: only for RSA key |
* |       |            | components]                                 |
* +-------+------------+---------------------------------------------+
* | Le    | 0x00       |                                             |
* +-------+------------+---------------------------------------------+
*
```

R-APDU Body

```
* +------------+------------------------------------------+
* | Value      | Description                              |
* +============+==========================================+
* | TLV[TAG_1] | Data read from the secure object.        |
* +------------+------------------------------------------+
*
```

R-APDU Trailer

```
* +-------------+--------------------------------+
* | SW          | Description                    |
* +=============+================================+
* | SW_NO_ERROR | The read is done successfully. |
* +-------------+--------------------------------+
*
```

Parameters

in	*session_ctx*	Session Context [0:kSE05x_pSession]
in	*objectID*	object id [1:kSE05x_TAG_1]
in	*offset*	offset [2:kSE05x_TAG_2]
in	*length*	length [3:kSE05x_TAG_3]
out	*data*	[0:kSE05x_TAG_1]
in,out	*pdataLen*	Length for data

2.3.2.9 Se05x_API_SessionClose()

```
smStatus_t Se05x_API_SessionClose (
            pSe05xSession_t session_ctx )
```

Se05x_API_SessionClose

Close session to SE05x.

Parameters

in	*session_ctx*	The session context

Returns

The sm status.

2.3.2.10 Se05x_API_SessionOpen()

```
smStatus_t Se05x_API_SessionOpen (
            pSe05xSession_t session_ctx )
```

Se05x_API_SessionOpen

Open session to SE05x. Multiple sessions are not supported.

Parameters

in	*session_ctx*	The session context

Returns

The sm status.

2.3.2.11 Se05x_API_WriteBinary()

```
smStatus_t Se05x_API_WriteBinary (
            pSe05xSession_t session_ctx,
            pSe05xPolicy_t policy,
            uint32_t objectID,
            uint16_t offset,
            uint16_t length,
            const uint8_t * inputData,
            size_t inputDataLen )
```

Se05x_API_WriteBinary

Creates or writes to a binary file object. Data are written to either the start of the file or (if specified) to the offset passed to the function.

Command to Applet

```
* +---------+-----------------+---------------------------------------------+
* | Field   | Value           | Description                                 |
* +=========+=================+=============================================+
* | P1      | P1_BINARY       | See :cpp:type:`SE05x_P1_t`                  |
* +---------+-----------------+---------------------------------------------+
* | P2      | P2_DEFAULT      | See :cpp:type:`SE05x_P2_t`                  |
* +---------+-----------------+---------------------------------------------+
* | Payload | TLV[TAG_POLICY] | Byte array containing the object policy.    |
* |         |                 | [Optional: default policy applies]          |
* |         |                 | [Conditional: only when the object identifier |
* |         |                 | is not in use yet]                          |
* +---------+-----------------+---------------------------------------------+
* |         | TLV[TAG_1]      | 4-byte object identifier                    |
* +---------+-----------------+---------------------------------------------+
* |         | TLV[TAG_2]      | 2-byte file offset   [Optional: default = 0]  |
* +---------+-----------------+---------------------------------------------+
* |         | TLV[TAG_3]      | 2-byte file length (up to 0x7FFF).          |
* |         |                 | [Conditional: only when the object identifier |
* |         |                 | is not in use yet]                          |
* +---------+-----------------+---------------------------------------------+
* |         | TLV[TAG_4]      | Data to be written   [Optional: if not given, |
* |         |                 | TAG_3 must be filled]                       |
* +---------+-----------------+---------------------------------------------+
* |         | TLV[TAG_11]     | 4-byte version    [Optional]                |
* +---------+-----------------+---------------------------------------------+
*
```

Parameters

in	*session_ctx*	Session Context [0:kSE05x_pSession]
in	*policy*	policy [1:kSE05x_TAG_POLICY]
in	*objectID*	object id [2:kSE05x_TAG_1]
in	*offset*	offset [3:kSE05x_TAG_2]
in	*length*	length [4:kSE05x_TAG_3]
in	*inputData*	input data. (Max - 128 Bytes) [5:kSE05x_TAG_4]
in	*inputDataLen*	Length of inputData

2.3.2.12 Se05x_API_WriteECKey()

```
smStatus_t Se05x_API_WriteECKey (
            pSe05xSession_t session_ctx,
            pSe05xPolicy_t policy,
            SE05x_MaxAttemps_t maxAttempt,
            uint32_t objectID,
            SE05x_ECCurve_t curveID,
            const uint8_t * privKey,
            size_t privKeyLen,
            const uint8_t * pubKey,
            size_t pubKeyLen,
            const SE05x_INS_t ins_type,
            const SE05x_KeyPart_t key_part )
```

Se05x_API_WriteECKey

Write or update an EC key object.

P1KeyPart indicates the key type to be created (if the object does not yet exist).

If P1KeyPart = P1_KEY_PAIR, Private Key Value (TLV[TAG_3]) and Public Key Value (TLV[TAG_4) must both be present, or both be absent. If absent, the key pair is generated in the SE05X .

If the object already exists, P1KeyPart is ignored.

```
* +---------+------------------------+---------------------------------------------+
* | Field   | Value                  | Description                                 |
* +=========+========================+=============================================+
* | P1      | :cpp:type:`SE05x_P1_t` | See  :cpp:type:`SE05x_P1_t` ,  P1KeyType     |
* |         | | P1_EC                | should only be set for new objects.         |
* +---------+------------------------+---------------------------------------------+
* | P2      | P2_DEFAULT             | See P2                                      |
* +---------+------------------------+---------------------------------------------+
* | Payload | TLV[TAG_POLICY]        | Byte array containing the object policy.    |
* |         |                        | [Optional: default policy applies]          |
* |         |                        | [Conditional - only when the object         |
* |         |                        | identifier is not in use yet]               |
* +---------+------------------------+---------------------------------------------+
* |         | TLV[TAG_MAX_ATTEMPTS]  | 2-byte maximum number of attempts. If 0 is  |
* |         |                        | given, this means unlimited.   [Optional:   |
* |         |                        | default unlimited]   [Conditional: only when |
* |         |                        | the object  identifier is not in use yet and |
* |         |                        | INS includes  INS_AUTH_OBJECT; see          |
* |         |                        | AuthenticationObjectPolicies ]              |
* +---------+------------------------+---------------------------------------------+
* |         | TLV[TAG_1]             | 4-byte object identifier                    |
* +---------+------------------------+---------------------------------------------+
* |         | TLV[TAG_2]             | 1-byte curve identifier, see ECCurve        |
* |         |                        | [Conditional: only when the object  identifier |
* |         |                        | is not in use yet; ]                        |
* +---------+------------------------+---------------------------------------------+
* |         | TLV[TAG_3]             | Private key value (see  :cpp:type:`ECKeyRef`  |
* |         |                        | )   [Conditional: only when the private key is |
* |         |                        | externally generated and P1KeyType is either |
* |         |                        | P1_KEY_PAIR  or P1_PRIVATE]                 |
* +---------+------------------------+---------------------------------------------+
* |         | TLV[TAG_4]             | Public key value (see  :cpp:type:`ECKeyRef`  ) |
* |         |                        | [Conditional: only when the public key is   |
* |         |                        | externally generated and P1KeyType is either |
* |         |                        | P1_KEY_PAIR  or P1_PUBLIC]                  |
* +---------+------------------------+---------------------------------------------+
* |         | TLV[TAG_11]            | 4-byte version    [Optional]                |
* +---------+------------------------+---------------------------------------------+
*
```

Parameters

in	*session_ctx*	The session context
in	*policy*	The policy
in	*maxAttempt*	The maximum attempt
in	*objectID*	The object id
in	*curveID*	The curve id
in	*privKey*	The priv key
in	*privKeyLen*	The priv key length
in	*pubKey*	The pub key
in	*pubKeyLen*	The pub key length
in	*ins_type*	The insert type
in	*key_part*	The key part

Returns

The sm status.

2.3.2.13 Se05x_API_WriteSymmKey()

```
smStatus_t Se05x_API_WriteSymmKey (
            pSe05xSession_t session_ctx,
            pSe05xPolicy_t policy,
            SE05x_MaxAttemps_t maxAttempt,
            uint32_t objectID,
            SE05x_KeyID_t kekID,
            const uint8_t * keyValue,
            size_t keyValueLen,
            const SE05x_INS_t ins_type,
            const SE05x_SymmKeyType_t type )
```

Se05x_API_WriteSymmKey

Creates or writes an AES key, DES key or HMAC key, indicated by P1:

- P1_AES
- P1_DES
- P1_HMAC

Users can pass RFC3394 wrapped keys by indicating the KEK in TLV[TAG_2]. Note that RFC3394 required 8-byte aligned input, so this can only be used when the key has an 8-byte aligned length.

Command to Applet

```
* +---------+-----------------------+-----------------------------------------------+
* | Field   | Value                 | Description                                   |
* +=========+=======================+===============================================+
* | P1      | See above             | See :cpp:type:`SE05x_P1_t`                    |
* +---------+-----------------------+-----------------------------------------------+
* | P2      | P2_DEFAULT            | See :cpp:type:`SE05x_P2_t`                    |
* +---------+-----------------------+-----------------------------------------------+
* | Payload | TLV[TAG_POLICY]       | Byte array containing the object policy.      |
* |         |                       | [Optional: default policy applies]            |
* |         |                       | [Conditional: only when the object identifier |
* |         |                       | is not in use yet]                            |
* +---------+-----------------------+-----------------------------------------------+
* |         | TLV[TAG_MAX_ATTEMPTS] | 2-byte maximum number of attempts. If 0 is    |
* |         |                       | given, this means unlimited.   [Optional:     |
* |         |                       | default unlimited]   [Conditional: only when  |
* |         |                       | the object identifier is not in use yet and   |
* |         |                       | INS includes  INS_AUTH_OBJECT; see            |
* |         |                       | AuthenticationObjectPolicies]                 |
* +---------+-----------------------+-----------------------------------------------+
* |         | TLV[TAG_1]            | 4-byte object identifier                      |
* +---------+-----------------------+-----------------------------------------------+
* |         | TLV[TAG_2]            | 4-byte KEK identifier   [Conditional: only    |
* |         |                       | when the key value is RFC3394 wrapped]        |
* +---------+-----------------------+-----------------------------------------------+
* |         | TLV[TAG_3]            | Key value, either plain or RFC3394 wrapped.   |
* +---------+-----------------------+-----------------------------------------------+
* |         | TLV[TAG_4]            | Tag length for GCM/GMAC. Will only be used if |
* |         |                       | the object is an  AESKey.   [Optional]        |
* +---------+-----------------------+-----------------------------------------------+
* |         | TLV[TAG_11]           | 4-byte version    [Optional]                  |
* +---------+-----------------------+-----------------------------------------------+
*
```

Parameters

in	*session_ctx*	The session context
in	*policy*	The policy
in	*maxAttempt*	The maximum attempt
in	*objectID*	The object id
in	*kekID*	The kek id
in	*keyValue*	The key value (Supported lengths - 128, 192 or 256 bits)
in	*keyValueLen*	The key value length
in	*ins_type*	The insert type
in	*type*	The type

Returns

The sm status.

2.4 se05x_APDU_apis.h

Go to the documentation of this file.

```
#ifndef SE05X_APDU_APIS_H_INC
#define SE05X_APDU_APIS_H_INC

/* ********************** Include files ********************** */
#include "se05x_types.h"
#include "se05x_tlv.h"

smStatus_t Se05x_API_SessionOpen(pSe05xSession_t session_ctx);

smStatus_t Se05x_API_SessionClose(pSe05xSession_t session_ctx);

smStatus_t Se05x_API_WriteECKey(pSe05xSession_t session_ctx,
    pSe05xPolicy_t policy,
    SE05x_MaxAttemps_t maxAttempt,
    uint32_t objectID,
    SE05x_ECCurve_t curveID,
    const uint8_t *privKey,
    size_t privKeyLen,
    const uint8_t *pubKey,
    size_t pubKeyLen,
    const SE05x_INS_t ins_type,
    const SE05x_KeyPart_t key_part);

smStatus_t Se05x_API_ReadObject(
    pSe05xSession_t session_ctx, uint32_t objectID, uint16_t offset, uint16_t length, uint8_t *data,
   size_t *pdataLen);

smStatus_t Se05x_API_GetVersion(pSe05xSession_t session_ctx, uint8_t *pappletVersion, size_t
   *appletVersionLen);

smStatus_t Se05x_API_ECDSASign(pSe05xSession_t session_ctx,
    uint32_t objectID,
    SE05x_ECSignatureAlgo_t ecSignAlgo,
    const uint8_t *inputData,
    size_t inputDataLen,
    uint8_t *signature,
    size_t *psignatureLen);

smStatus_t Se05x_API_ECDSAVerify(pSe05xSession_t session_ctx,
    uint32_t objectID,
    SE05x_ECSignatureAlgo_t ecSignAlgo,
    const uint8_t *inputData,
    size_t inputDataLen,
    const uint8_t *signature,
    size_t signatureLen,
    SE05x_Result_t *presult);

smStatus_t Se05x_API_CheckObjectExists(pSe05xSession_t session_ctx, uint32_t objectID, SE05x_Result_t
   *presult);

```

```
smStatus_t Se05x_API_WriteBinary(pSe05xSession_t session_ctx,
    pSe05xPolicy_t policy,
    uint32_t objectID,
    uint16_t offset,
    uint16_t length,
    const uint8_t *inputData,
    size_t inputDataLen);

smStatus_t Se05x_API_ECDHGenerateSharedSecret(pSe05xSession_t session_ctx,
    uint32_t objectID,
    const uint8_t *pubKey,
    size_t pubKeyLen,
    uint8_t *sharedSecret,
    size_t *psharedSecretLen);

smStatus_t Se05x_API_CipherOneShot(pSe05xSession_t session_ctx,
    uint32_t objectID,
    SE05x_CipherMode_t cipherMode,
    const uint8_t *inputData,
    size_t inputDataLen,
    uint8_t *IV,
    size_t IVLen,
    uint8_t *outputData,
    size_t *poutputDataLen,
    const SE05x_Cipher_Oper_OneShot_t operation);

smStatus_t Se05x_API_WriteSymmKey(pSe05xSession_t session_ctx,
    pSe05xPolicy_t policy,
    SE05x_MaxAttemps_t maxAttempt,
    uint32_t objectID,
    SE05x_KeyID_t kekID,
    const uint8_t *keyValue,
    size_t keyValueLen,
    const SE05x_INS_t ins_type,
    const SE05x_SymmKeyType_t type);

smStatus_t Se05x_API_DeleteSecureObject(pSe05xSession_t session_ctx, uint32_t objectID);

#endif //#ifndef SE05X_APDU_APIS_H_INC
```

Index