

i.MX8 HSM API

Revision_0.1

Generated by Doxygen 1.8.15

1 Main Page	1
2 Revision History	1
3 General concepts related to the API	2
3.1 Session	2
3.2 Service flow	2
4 Module Index	2
4.1 Modules	2
5 Class Index	2
5.1 Class List	2
6 Module Documentation	3
6.1 Hsm_api	3
6.1.1 Detailed Description	7
6.1.2 Macro Definition Documentation	7
6.1.3 Typedef Documentation	13
6.1.4 Enumeration Type Documentation	16
6.1.5 Function Documentation	17
7 Class Documentation	30
7.1 hsm_op_ecies_dec_args_t Struct Reference	30
7.1.1 Member Data Documentation	31
7.2 hsm_op_ecies_enc_args_t Struct Reference	33
7.2.1 Member Data Documentation	34
7.3 hsm_op_pub_key_dec_args_t Struct Reference	37
7.3.1 Detailed Description	37
7.3.2 Member Data Documentation	37
7.4 hsm_op_pub_key_rec_args_t Struct Reference	39
7.4.1 Member Data Documentation	40
7.5 op_butl_key_exp_args_t Struct Reference	42
7.5.1 Member Data Documentation	42
7.6 op_cipher_one_go_args_t Struct Reference	45
7.6.1 Member Data Documentation	45
7.7 op_finalize_sign_args_t Struct Reference	47
7.7.1 Member Data Documentation	47
7.8 op_generate_key_args_t Struct Reference	48
7.8.1 Member Data Documentation	49
7.9 op_generate_sign_args_t Struct Reference	50
7.9.1 Member Data Documentation	50
7.10 op_get_random_args_t Struct Reference	52
7.10.1 Member Data Documentation	52

7.11 op_hash_one_go_args_t Struct Reference	52
7.11.1 Member Data Documentation	52
7.12 op_import_public_key_args_t Struct Reference	54
7.12.1 Member Data Documentation	54
7.13 op_manage_key_args_t Struct Reference	55
7.13.1 Member Data Documentation	55
7.14 op_prepare_sign_args_t Struct Reference	56
7.14.1 Member Data Documentation	56
7.15 op_verify_sign_args_t Struct Reference	57
7.15.1 Member Data Documentation	57
7.16 open_session_args_t Struct Reference	59
7.16.1 Member Data Documentation	59
7.17 open_svc_cipher_args_t Struct Reference	60
7.17.1 Member Data Documentation	60
7.18 open_svc_hash_args_t Struct Reference	61
7.18.1 Member Data Documentation	61
7.19 open_svc_key_management_args_t Struct Reference	62
7.19.1 Member Data Documentation	62
7.20 open_svc_key_store_args_t Struct Reference	63
7.20.1 Member Data Documentation	63
7.21 open_svc_rng_args_t Struct Reference	64
7.21.1 Member Data Documentation	64
7.22 open_svc_sign_gen_args_t Struct Reference	65
7.22.1 Member Data Documentation	65
7.23 open_svc_sign_ver_args_t Struct Reference	66
7.23.1 Member Data Documentation	66
Index	67

1 Main Page

This document is a software referece description of the API provided by the i.MX8 HSM solutions.

2 Revision History

Revision 0.1: 29/03/2019 Savari preliminary draft - subject to change
Revision 0.8: 20/05/2019 Adding butterfly key expansion operation; adding signature, rng, hash services.

3 General concepts related to the API

3.1 Session

The API must be initialized by a potential requestor by opening a session.

The session establishes a route (MU, DomainID...) between the requestor and the HSM, and grants the usage of a specified key store through a password authentication.

When a session is opened, the HSM returns a handle identifying the session to the requestor.

3.2 Service flow

For a given category of services, the requestor is expected to open a service flow by invoking the appropriate HSM API. The session handle, as well as the control data needed for the service flow are provided as parameters of the call. Upon reception of the open request, the HSM allocates a context in which the session handle, as well as the provided control parameters are stored. The context is preserved until the service flow is closed by the user and it is used by the HSM to proceed with the sub-sequent operations requested by the user on the service flow.

4 Module Index

4.1 Modules

Here is a list of all modules:

Hsm_api	3
----------------	----------

5 Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

hsm_op_ecies_dec_args_t	30
hsm_op_ecies_enc_args_t	33
hsm_op_pub_key_dec_args_t	37
hsm_op_pub_key_rec_args_t	39
op_but_key_exp_args_t	42
op_cipher_one_go_args_t	45
op_finalize_sign_args_t	47
op_generate_key_args_t	48
op_generate_sign_args_t	50

op_get_random_args_t	52
op_hash_one_go_args_t	52
op_import_public_key_args_t	54
op_manage_key_args_t	55
op_prepare_sign_args_t	56
op_verify_sign_args_t	57
open_session_args_t	59
open_svc_cipher_args_t	60
open_svc_hash_args_t	61
open_svc_key_management_args_t	62
open_svc_key_store_args_t	63
open_svc_rng_args_t	64
open_svc_sign_gen_args_t	65
open_svc_sign_ver_args_t	66

6 Module Documentation

6.1 Hsm_api

i.MX8 HSM API header file

Classes

- struct [open_session_args_t](#)
- struct [open_svc_key_store_args_t](#)
- struct [open_svc_key_management_args_t](#)
- struct [op_generate_key_args_t](#)
- struct [op_manage_key_args_t](#)
- struct [op_butl_key_exp_args_t](#)
- struct [open_svc_cipher_args_t](#)
- struct [op_cipher_one_go_args_t](#)
- struct [hsm_op_ecies_dec_args_t](#)
- struct [open_svc_sign_gen_args_t](#)
- struct [op_generate_sign_args_t](#)
- struct [op_prepare_sign_args_t](#)
- struct [op_finalize_sign_args_t](#)
- struct [open_svc_sign_ver_args_t](#)
- struct [op_verify_sign_args_t](#)
- struct [op_import_public_key_args_t](#)
- struct [open_svc_rng_args_t](#)
- struct [op_get_random_args_t](#)
- struct [open_svc_hash_args_t](#)
- struct [op_hash_one_go_args_t](#)
- struct [hsm_op_pub_key_rec_args_t](#)
- struct [hsm_op_pub_key_dec_args_t](#)
- struct [hsm_op_ecies_enc_args_t](#)

Macros

- #define HSM_SVC_KEY_STORE_FLAGS_CREATE ((hsm_svc_key_store_flags_t)(1 << 0))
- #define HSM_SVC_KEY_STORE_FLAGS_UPDATE ((hsm_svc_key_store_flags_t)(1 << 1))
- #define HSM_SVC_KEY_STORE_FLAGS_DELETE ((hsm_svc_key_store_flags_t)(1 << 3))
- #define HSM_KEY_TYPE_ECDSA_NIST_P224 ((hsm_key_type_t)0x01)
- #define HSM_KEY_TYPE_ECDSA_NIST_P256 ((hsm_key_type_t)0x02)
- #define HSM_KEY_TYPE_ECDSA_NIST_P384 ((hsm_key_type_t)0x03)
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_224 ((hsm_key_type_t)0x12)
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_256 ((hsm_key_type_t)0x13)
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_384 ((hsm_key_type_t)0x15)
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_224 ((hsm_key_type_t)0x22)
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_256 ((hsm_key_type_t)0x23)
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_384 ((hsm_key_type_t)0x25)
- #define HSM_KEY_TYPE_AES_128 ((hsm_key_type_t)0x30)
- #define HSM_KEY_TYPE_AES_192 ((hsm_key_type_t)0x31)
- #define HSM_KEY_TYPE_AES_256 ((hsm_key_type_t)0x32)
- #define HSM_KEY_INFO_PERMANENT ((hsm_key_info_t)(1 << 0))
- #define HSM_OP_KEY_GENERATION_FLAGS_UPDATE ((hsm_op_key_gen_flags_t)(1 << 0))
- #define HSM_OP_KEY_GENERATION_FLAGS_CREATE_PERSISTENT ((hsm_op_key_gen_flags_t)(1 << 1))
- #define HSM_OP_KEY_GENERATION_FLAGS_CREATE_TRANSIENT ((hsm_op_key_gen_flags_t)(1 << 2))
- #define HSM_OP_KEY_GENERATION_FLAGS_STRICT_OPERATION ((hsm_op_key_gen_flags_t)(1 << 7))
- #define HSM_OP_MANAGE_KEY_FLAGS_UPDATE ((hsm_op_manage_key_flags_t)(1 << 0))
- #define HSM_OP_MANAGE_KEY_FLAGS_CREATE_PERSISTENT ((hsm_op_manage_key_flags_t)(1 << 1))
- #define HSM_OP_MANAGE_KEY_FLAGS_CREATE_TRANSIENT ((hsm_op_manage_key_flags_t)(1 << 2))
- #define HSM_OP_MANAGE_KEY_FLAGS_DELETE ((hsm_op_manage_key_flags_t)(1 << 3))
- #define HSM_OP_MANAGE_KEY_FLAGS_STRICT_OPERATION ((hsm_op_manage_key_flags_t)(1 << 7))
- #define HSM_CIPHER_ONE_GO_ALGO_AES_ECB ((hsm_op_cipher_one_go_algo_t)(0x00))
- #define HSM_CIPHER_ONE_GO_ALGO_AES_CBC ((hsm_op_cipher_one_go_algo_t)(0x01))
- #define HSM_CIPHER_ONE_GO_ALGO_AES_CCM ((hsm_op_cipher_one_go_algo_t)(0x04))

AES CCM where Adata = 0, Tlen = 16 bytes.

- #define HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT ((hsm_op_cipher_one_go_flags_t)(1 << 0))
- #define HSM_CIPHER_ONE_GO_FLAGS_DECRYPT ((hsm_op_cipher_one_go_flags_t)(1 << 1))
- #define HSM_OP_GENERATE_SIGN_INPUT_DIGEST ((hsm_op_generate_sign_flags_t)(1 << 0))
- #define HSM_OP_GENERATE_SIGN_INPUT_MESSAGE ((hsm_op_generate_sign_flags_t)(1 << 1))
- #define HSM_OP_GENERATE_SIGN_COMPRESSED_POINT ((hsm_op_generate_sign_flags_t)(1 << 2))
- #define HSM_SIGNATURE_SCHEME_ECDSA_NIST_P224_SHA_256 ((hsm_signature_scheme_id_t)0x01)
- #define HSM_SIGNATURE_SCHEME_ECDSA_NIST_P256_SHA_256 ((hsm_signature_scheme_id_t)0x02)
- #define HSM_SIGNATURE_SCHEME_ECDSA_NIST_P384_SHA_384 ((hsm_signature_scheme_id_t)0x03)
- #define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_224_SHA_256 ((hsm_signature_scheme_id_t)0x12)
- #define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_256_SHA_256 ((hsm_signature_scheme_id_t)0x13)
- #define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_384_SHA_384 ((hsm_signature_scheme_id_t)0x15)
- #define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_224_SHA_256 ((hsm_signature_scheme_id_t)0x22)
- #define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_256_SHA_256 ((hsm_signature_scheme_id_t)0x23)
- #define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_384_SHA_384 ((hsm_signature_scheme_id_t)0x25)
- #define HSM_OP_FINALIZE_SIGN_INPUT_DIGEST ((hsm_op_finalize_sign_flags_t)(1 << 0))
- #define HSM_OP_FINALIZE_SIGN_INPUT_MESSAGE ((hsm_op_finalize_sign_flags_t)(1 << 1))
- #define HSM_OP_FINALIZE_SIGN_COMPRESSED_POINT ((hsm_op_finalize_sign_flags_t)(1 << 2))
- #define HSM_OP_VERIFY_SIGN_INPUT_DIGEST ((hsm_op_verify_sign_flags_t)(1 << 0))

- `#define HSM_OP_VERIFY_SIGN_INPUT_MESSAGE ((hsm_op_verify_sign_flags_t)(1 << 1))`
- `#define HSM_OP_VERIFY_SIGN_COMPRESSED_POINT ((hsm_op_verify_sign_flags_t)(1 << 2))`
- `#define HSM_OP_VERIFY_SIGN_KEY_INTERNAL ((hsm_op_verify_sign_flags_t)(1 << 4))`
- `#define HSM_VERIFICATION_STATUS_SUCCESS ((hsm_verification_status_t)(0x5A3CC3A5))`
- `#define HSM_HASH_ALGO_SHA_224 ((hsm_hash_algo_t)(0x0))`
- `#define HSM_HASH_ALGO_SHA_256 ((hsm_hash_algo_t)(0x1))`
- `#define HSM_HASH_ALGO_SHA_384 ((hsm_hash_algo_t)(0x2))`

Typedefs

- `typedef uint32_t hsm_hdl_t`
- `typedef uint8_t hsm_svc_key_store_flags_t`
- `typedef uint8_t hsm_svc_key_management_flags_t`
- `typedef uint8_t hsm_svc_cipher_flags_t`
- `typedef uint8_t hsm_svc_signature_generation_flags_t`
- `typedef uint8_t hsm_svc_signature_verification_flags_t`
- `typedef uint8_t hsm_svc_fast_signature_verification_flags_t`
- `typedef uint8_t hsm_svc_rng_flags_t`
- `typedef uint8_t hsm_svc_hash_flags_t`
- `typedef uint8_t hsm_op_key_gen_flags_t`
- `typedef uint8_t hsm_op_manage_key_flags_t`
- `typedef uint8_t hsm_op_but_key_exp_flags_t`
- `typedef uint8_t hsm_op_cipher_one_go_algo_t`
- `typedef uint8_t hsm_op_cipher_one_go_flags_t`
- `typedef uint8_t hsm_op_generate_sign_flags_t`
- `typedef uint8_t hsm_op_prepare_signature_flags_t`
- `typedef uint8_t hsm_op_finalize_sign_flags_t`
- `typedef uint8_t hsm_op_verify_sign_flags_t`
- `typedef uint8_t hsm_op_hash_one_go_flags_t`
- `typedef uint8_t hsm_op_pub_key_rec_flags_t`
- `typedef uint8_t hsm_op_pub_key_dec_flags_t`
- `typedef uint8_t hsm_op_ecies_enc_flags_t`
- `typedef uint8_t hsm_op_ecies_dec_flags_t`
- `typedef uint8_t hsm_signature_scheme_id_t`
- `typedef uint8_t hsm_hash_algo_t`
- `typedef uint8_t hsm_key_type_t`
- `typedef uint8_t hsm_key_type_ext_t`
- `typedef uint16_t hsm_key_info_t`
- `typedef uint32_t hsm_addr_msb_t`
- `typedef uint32_t hsm_addr_lsb_t`
- `typedef uint32_t hsm_verification_status_t`

Enumerations

- `enum hsm_err_t {`
`HSM_NO_ERROR = 0x0,`
`HSM_INVALID_MESSAGE = 0x1,`
`HSM_INVALID_ADDRESS = 0x2,`
`HSM_UNKNOWN_ID = 0x3,`
`HSM_INVALID_PARAM = 0x4,`
`HSM_NVM_ERROR = 0x5,`
`HSM_OUT_OF_MEMORY = 0x6,`
`HSM_UNKNOWN_HANDLE = 0x7,`
`}`

```

HSM_UNKNOWN_KEY_STORE = 0x8,
HSM_KEY_STORE_AUTH = 0x9,
HSM_KEY_STORAGE_ERROR = 0xA,
HSM_ID_CONFLICT = 0xB,
HSM_RNG_NOT_STARTED = 0xC,
HSM_CMD_NOT_SUPPORTED = 0xD,
HSM_INVALID_LIFECYCLE = 0xE,
HSM_KEY_STORE_CONFLICT = 0xF,
HSM_GENERAL_ERROR = 0xFF }

```

Error codes returned by HSM functions.

Functions

- `hsm_err_t hsm_open_session (open_session_args_t *args, hsm_hdl_t *session_hdl)`
- `hsm_err_t hsm_close_session (hsm_hdl_t session_hdl)`
- `hsm_err_t hsm_open_key_store_service (hsm_hdl_t session_hdl, open_svc_key_store_args_t *args, hsm_hdl_t *key_store_hdl)`
- `hsm_err_t hsm_close_key_store_service (hsm_hdl_t key_store_hdl)`
- `hsm_err_t hsm_open_key_management_service (hsm_hdl_t key_store_hdl, open_svc_key_management_args_t *args, hsm_hdl_t *key_management_hdl)`
- `hsm_err_t hsm_generate_key (hsm_hdl_t key_management_hdl, op_generate_key_args_t args)`
- `hsm_err_t hsm_manage_key (hsm_hdl_t key_management_hdl, op_manage_key_args_t *args)`
- `hsm_err_t hsm_butterfly_key_expansion (hsm_hdl_t key_management_hdl, op_butt_key_exp_args_t *args)`
- `hsm_err_t hsm_close_key_management_service (hsm_hdl_t key_management_hdl)`
- `hsm_err_t hsm_open_cipher_service (hsm_hdl_t key_store_hdl, open_svc_cipher_args_t *args, hsm_hdl_t *cipher_hdl)`
- `hsm_err_t hsm_cipher_one_go (hsm_hdl_t cipher_hdl, op_cipher_one_go_args_t *args)`
- `hsm_err_t hsm_ecies_decryption (hsm_hdl_t cipher_hdl, hsm_op_ecies_dec_args_t *args)`
- `hsm_err_t hsm_close_cipher_service (hsm_hdl_t cipher_hdl)`
- `hsm_err_t hsm_open_signature_generation_service (hsm_hdl_t key_store_hdl, open_svc_sign_gen_args_t *args, hsm_hdl_t *signature_gen_hdl)`
- `hsm_err_t hsm_close_signature_generation_service (hsm_hdl_t signature_gen_hdl)`
- `hsm_err_t hsm_generate_signature (hsm_hdl_t signature_gen_hdl, op_generate_sign_args_t *args)`
- `hsm_err_t hsm_prepare_signature (hsm_hdl_t signature_gen_hdl, op_prepare_sign_args_t *args)`
- `hsm_err_t hsm_finalize_signature (hsm_hdl_t signature_gen_hdl, op_finalize_sign_args_t *args)`
- `hsm_err_t hsm_open_signature_verification_service (hsm_hdl_t session_hdl, open_svc_sign_ver_args_t *args, hsm_hdl_t *signature_ver_hdl)`
- `hsm_err_t hsm_verify_signature (hsm_hdl_t signature_ver_hdl, op_verify_sign_args_t *args, hsm_verification_status_t *status)`
- `hsm_err_t hsm_import_public_key (hsm_hdl_t signature_ver_hdl, op_import_public_key_args_t *args, hsm_addr_lsb_t *int_key)`
- `hsm_err_t hsm_close_signature_verification_service (hsm_hdl_t signature_ver_hdl)`
- `hsm_err_t hsm_open_rng_service (hsm_hdl_t session_hdl, open_svc_rng_args_t *args, hsm_hdl_t *rng_hdl)`
- `hsm_err_t hsm_close_rng_service (hsm_hdl_t rng_hdl)`
- `hsm_err_t hsm_get_random (hsm_hdl_t rng_hdl, op_get_random_args_t *args)`
- `hsm_err_t hsm_open_hash_service (hsm_hdl_t session_hdl, open_svc_hash_args_t *args, hsm_hdl_t *hash_hdl)`
- `hsm_err_t hsm_close_hash_service (hsm_hdl_t hash_hdl)`
- `hsm_err_t hsm_hash_one_go (hsm_hdl_t hash_hdl, op_hash_one_go_args_t *args)`
- `hsm_err_t hsm_pub_key_reconstruction (hsm_hdl_t session_hdl, hsm_op_pub_key_rec_args_t *args)`
- `hsm_err_t hsm_pub_key_decompression (hsm_hdl_t session_hdl, hsm_op_pub_key_dec_args_t *args)`
- `hsm_err_t hsm_ecies_encryption (hsm_hdl_t session_hdl, hsm_op_ecies_enc_args_t *args)`

6.1.1 Detailed Description

i.MX8 HSM API header file

6.1.2 Macro Definition Documentation

6.1.2.1 HSM_SVC_KEY_STORE_FLAGS_CREATE

```
#define HSM_SVC_KEY_STORE_FLAGS_CREATE ((hsm_svc_key_store_flags_t) (1 << 0))
```

It must be specified to create a new key storage

6.1.2.2 HSM_SVC_KEY_STORE_FLAGS_UPDATE

```
#define HSM_SVC_KEY_STORE_FLAGS_UPDATE ((hsm_svc_key_store_flags_t) (1 << 1))
```

6.1.2.3 HSM_SVC_KEY_STORE_FLAGS_DELETE

```
#define HSM_SVC_KEY_STORE_FLAGS_DELETE ((hsm_svc_key_store_flags_t) (1 << 3))
```

6.1.2.4 HSM_KEY_TYPE_ECDSA_NIST_P224

```
#define HSM_KEY_TYPE_ECDSA_NIST_P224 ((hsm_key_type_t) 0x01)
```

6.1.2.5 HSM_KEY_TYPE_ECDSA_NIST_P256

```
#define HSM_KEY_TYPE_ECDSA_NIST_P256 ((hsm_key_type_t) 0x02)
```

6.1.2.6 HSM_KEY_TYPE_ECDSA_NIST_P384

```
#define HSM_KEY_TYPE_ECDSA_NIST_P384 ((hsm_key_type_t) 0x03)
```

6.1.2.7 HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_224

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_224 ((hsm_key_type_t) 0x12)
```

6.1.2.8 HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_256

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_256 ((hsm_key_type_t)0x13)
```

6.1.2.9 HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_384

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_384 ((hsm_key_type_t)0x15)
```

6.1.2.10 HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_224

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_224 ((hsm_key_type_t)0x22)
```

6.1.2.11 HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_256

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_256 ((hsm_key_type_t)0x23)
```

6.1.2.12 HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_384

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_384 ((hsm_key_type_t)0x25)
```

6.1.2.13 HSM_KEY_TYPE_AES_128

```
#define HSM_KEY_TYPE_AES_128 ((hsm_key_type_t)0x30)
```

6.1.2.14 HSM_KEY_TYPE_AES_192

```
#define HSM_KEY_TYPE_AES_192 ((hsm_key_type_t)0x31)
```

6.1.2.15 HSM_KEY_TYPE_AES_256

```
#define HSM_KEY_TYPE_AES_256 ((hsm_key_type_t)0x32)
```

6.1.2.16 HSM_KEY_INFO_PERMANENT

```
#define HSM_KEY_INFO_PERMANENT ((hsm_key_info_t)(1 << 0))
```

When set, the key is permanent. Once created, it will not be possible to update or delete the key anymore. This bit can never be reset.

6.1.2.17 HSM_OP_KEY_GENERATION_FLAGS_UPDATE

```
#define HSM_OP_KEY_GENERATION_FLAGS_UPDATE ((hsm_op_key_gen_flags_t) (1 << 0))
```

User can replace an existing key only by generating a key with the same type of the original one.

6.1.2.18 HSM_OP_KEY_GENERATION_FLAGS_CREATE_PERSISTENT

```
#define HSM_OP_KEY_GENERATION_FLAGS_CREATE_PERSISTENT ((hsm_op_key_gen_flags_t) (1 << 1))
```

Persistent keys are saved in the non volatile memory.

6.1.2.19 HSM_OP_KEY_GENERATION_FLAGS_CREATE_TRANSIENT

```
#define HSM_OP_KEY_GENERATION_FLAGS_CREATE_TRANSIENT ((hsm_op_key_gen_flags_t) (1 << 2))
```

Transient keys are deleted when the corresponding key store service flow is closed.

6.1.2.20 HSM_OP_KEY_GENERATION_FLAGS_STRICT_OPERATION

```
#define HSM_OP_KEY_GENERATION_FLAGS_STRICT_OPERATION ((hsm_op_key_gen_flags_t) (1 << 7))
```

The request is completed only when the new key has been written in the NVM. This applicable for persistent key only.

6.1.2.21 HSM_OP_MANAGE_KEY_FLAGS_UPDATE

```
#define HSM_OP_MANAGE_KEY_FLAGS_UPDATE ((hsm_op_manage_key_flags_t) (1 << 0))
```

User can replace an existing key only by importing a key with the same type of the original one.

6.1.2.22 HSM_OP_MANAGE_KEY_FLAGS_CREATE_PERSISTENT

```
#define HSM_OP_MANAGE_KEY_FLAGS_CREATE_PERSISTENT ((hsm_op_manage_key_flags_t) (1 << 1))
```

Persistent keys are saved in the non volatile memory.

6.1.2.23 HSM_OP_MANAGE_KEY_FLAGS_CREATE_TRANSIENT

```
#define HSM_OP_MANAGE_KEY_FLAGS_CREATE_TRANSIENT ((hsm_op_manage_key_flags_t) (1 << 2))
```

Transient keys are deleted when the corresponding key store service flow is closed.

6.1.2.24 HSM_OP_MANAGE_KEY_FLAGS_DELETE

```
#define HSM_OP_MANAGE_KEY_FLAGS_DELETE ((hsm_op_manage_key_flags_t) (1 << 3))
```

delete an existing key

6.1.2.25 HSM_OP_MANAGE_KEY_FLAGS_STRICT_OPERATION

```
#define HSM_OP_MANAGE_KEY_FLAGS_STRICT_OPERATION ((hsm_op_manage_key_flags_t) (1 << 7))
```

The request is completed only when the new key has been written in the NVM. This applicable for persistent key only.

6.1.2.26 HSM_CIPHER_ONE_GO_ALGO_AES_ECB

```
#define HSM_CIPHER_ONE_GO_ALGO_AES_ECB ((hsm_op_cipher_one_go_algo_t) (0x00))
```

6.1.2.27 HSM_CIPHER_ONE_GO_ALGO_AES_CBC

```
#define HSM_CIPHER_ONE_GO_ALGO_AES_CBC ((hsm_op_cipher_one_go_algo_t) (0x01))
```

6.1.2.28 HSM_CIPHER_ONE_GO_ALGO_AES_CCM

```
#define HSM_CIPHER_ONE_GO_ALGO_AES_CCM ((hsm_op_cipher_one_go_algo_t) (0x04))
```

AES CCM where Adata = 0, Tlen = 16 bytes.

Perform AES CCM with following constraints:

- Adata = 0 - There is no associated data
- Tlen = 16 bytes

6.1.2.29 HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT

```
#define HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT ((hsm_op_cipher_one_go_flags_t) (1 << 0))
```

6.1.2.30 HSM_CIPHER_ONE_GO_FLAGS_DECRYPT

```
#define HSM_CIPHER_ONE_GO_FLAGS_DECRYPT ((hsm_op_cipher_one_go_flags_t) (1 << 1))
```

6.1.2.31 HSM_OP_GENERATE_SIGN_INPUT_DIGEST

```
#define HSM_OP_GENERATE_SIGN_INPUT_DIGEST ((hsm_op_generate_sign_flags_t) (1 << 0))
```

6.1.2.32 HSM_OP_GENERATE_SIGN_INPUT_MESSAGE

```
#define HSM_OP_GENERATE_SIGN_INPUT_MESSAGE ((hsm_op_generate_sign_flags_t) (1 << 1))
```

6.1.2.33 HSM_OP_GENERATE_SIGN_COMPRESSED_POINT

```
#define HSM_OP_GENERATE_SIGN_COMPRESSED_POINT ((hsm_op_generate_sign_flags_t) (1 << 2))
```

6.1.2.34 HSM_SIGNATURE_SCHEME_ECDSA_NIST_P224_SHA_256

```
#define HSM_SIGNATURE_SCHEME_ECDSA_NIST_P224_SHA_256 ((hsm_signature_scheme_id_t) 0x01)
```

6.1.2.35 HSM_SIGNATURE_SCHEME_ECDSA_NIST_P256_SHA_256

```
#define HSM_SIGNATURE_SCHEME_ECDSA_NIST_P256_SHA_256 ((hsm_signature_scheme_id_t) 0x02)
```

6.1.2.36 HSM_SIGNATURE_SCHEME_ECDSA_NIST_P384_SHA_384

```
#define HSM_SIGNATURE_SCHEME_ECDSA_NIST_P384_SHA_384 ((hsm_signature_scheme_id_t) 0x03)
```

6.1.2.37 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_224_SHA_256

```
#define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_224_SHA_256 ((hsm_signature_scheme_id_t) 0x12)
```

6.1.2.38 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_256_SHA_256

```
#define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_256_SHA_256 ((hsm_signature_scheme_id_t) 0x13)
```

6.1.2.39 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_384_SHA_384

```
#define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_384_SHA_384 ((hsm_signature_scheme_id_t) 0x15)
```

6.1.2.40 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_224_SHA_256

```
#define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_224_SHA_256 ((hsm_signature_scheme_id_t) 0x22)
```

6.1.2.41 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_256_SHA_256

```
#define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_256_SHA_256 ((hsm_signature_scheme_id_t)0x23)
```

6.1.2.42 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_384_SHA_384

```
#define HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_384_SHA_384 ((hsm_signature_scheme_id_t)0x25)
```

6.1.2.43 HSM_OP_FINALIZE_SIGN_INPUT_DIGEST

```
#define HSM_OP_FINALIZE_SIGN_INPUT_DIGEST ((hsm_op_finalize_sign_flags_t)(1 << 0))
```

6.1.2.44 HSM_OP_FINALIZE_SIGN_INPUT_MESSAGE

```
#define HSM_OP_FINALIZE_SIGN_INPUT_MESSAGE ((hsm_op_finalize_sign_flags_t)(1 << 1))
```

6.1.2.45 HSM_OP_FINALIZE_SIGN_COMPRESSED_POINT

```
#define HSM_OP_FINALIZE_SIGN_COMPRESSED_POINT ((hsm_op_finalize_sign_flags_t)(1 << 2))
```

6.1.2.46 HSM_OP_VERIFY_SIGN_INPUT_DIGEST

```
#define HSM_OP_VERIFY_SIGN_INPUT_DIGEST ((hsm_op_verify_sign_flags_t)(1 << 0))
```

6.1.2.47 HSM_OP_VERIFY_SIGN_INPUT_MESSAGE

```
#define HSM_OP_VERIFY_SIGN_INPUT_MESSAGE ((hsm_op_verify_sign_flags_t)(1 << 1))
```

6.1.2.48 HSM_OP_VERIFY_SIGN_COMPRESSED_POINT

```
#define HSM_OP_VERIFY_SIGN_COMPRESSED_POINT ((hsm_op_verify_sign_flags_t)(1 << 2))
```

6.1.2.49 HSM_OP_VERIFY_SIGN_KEY_INTERNAL

```
#define HSM_OP_VERIFY_SIGN_KEY_INTERNAL ((hsm_op_verify_sign_flags_t)(1 << 4))
```

when set the value passed by the key argument is considered as the internal reference of a key imported through the `hsm__import_pub_key` API.

6.1.2.50 HSM_VERIFICATION_STATUS_SUCCESS

```
#define HSM_VERIFICATION_STATUS_SUCCESS ((hsm_verification_status_t) (0x5A3CC3A5))
```

6.1.2.51 HSM_HASH_ALGO_SHA_224

```
#define HSM_HASH_ALGO_SHA_224 ((hsm_hash_algo_t) (0x0))
```

6.1.2.52 HSM_HASH_ALGO_SHA_256

```
#define HSM_HASH_ALGO_SHA_256 ((hsm_hash_algo_t) (0x1))
```

6.1.2.53 HSM_HASH_ALGO_SHA_384

```
#define HSM_HASH_ALGO_SHA_384 ((hsm_hash_algo_t) (0x2))
```

6.1.3 Typedef Documentation

6.1.3.1 hsm_hdl_t

```
typedef uint32_t hsm_hdl_t
```

6.1.3.2 hsm_svc_key_store_flags_t

```
typedef uint8_t hsm_svc_key_store_flags_t
```

6.1.3.3 hsm_svc_key_management_flags_t

```
typedef uint8_t hsm_svc_key_management_flags_t
```

6.1.3.4 hsm_svc_cipher_flags_t

```
typedef uint8_t hsm_svc_cipher_flags_t
```

6.1.3.5 hsm_svc_signature_generation_flags_t

```
typedef uint8_t hsm_svc_signature_generation_flags_t
```

6.1.3.6 hsm_svc_signature_verification_flags_t

```
typedef uint8_t hsm_svc_signature_verification_flags_t
```

6.1.3.7 hsm_svc_fast_signature_verification_flags_t

```
typedef uint8_t hsm_svc_fast_signature_verification_flags_t
```

6.1.3.8 hsm_svc_rng_flags_t

```
typedef uint8_t hsm_svc_rng_flags_t
```

6.1.3.9 hsm_svc_hash_flags_t

```
typedef uint8_t hsm_svc_hash_flags_t
```

6.1.3.10 hsm_op_key_gen_flags_t

```
typedef uint8_t hsm_op_key_gen_flags_t
```

6.1.3.11 hsm_op_manage_key_flags_t

```
typedef uint8_t hsm_op_manage_key_flags_t
```

6.1.3.12 hsm_op_but_key_exp_flags_t

```
typedef uint8_t hsm_op_but_key_exp_flags_t
```

6.1.3.13 hsm_op_cipher_one_go_algo_t

```
typedef uint8_t hsm_op_cipher_one_go_algo_t
```


6.1.3.14 hsm_op_cipher_one_go_flags_t

```
typedef uint8_t hsm_op_cipher_one_go_flags_t
```

6.1.3.15 hsm_op_generate_sign_flags_t

```
typedef uint8_t hsm_op_generate_sign_flags_t
```

6.1.3.16 hsm_op_prepare_signature_flags_t

```
typedef uint8_t hsm_op_prepare_signature_flags_t
```

6.1.3.17 hsm_op_finalize_sign_flags_t

```
typedef uint8_t hsm_op_finalize_sign_flags_t
```

6.1.3.18 hsm_op_verify_sign_flags_t

```
typedef uint8_t hsm_op_verify_sign_flags_t
```

6.1.3.19 hsm_op_hash_one_go_flags_t

```
typedef uint8_t hsm_op_hash_one_go_flags_t
```

6.1.3.20 hsm_op_pub_key_rec_flags_t

```
typedef uint8_t hsm_op_pub_key_rec_flags_t
```

6.1.3.21 hsm_op_pub_key_dec_flags_t

```
typedef uint8_t hsm_op_pub_key_dec_flags_t
```

6.1.3.22 hsm_op_ecies_enc_flags_t

```
typedef uint8_t hsm_op_ecies_enc_flags_t
```

6.1.3.23 hsm_op_ecies_dec_flags_t

```
typedef uint8_t hsm_op_ecies_dec_flags_t
```

6.1.3.24 hsm_signature_scheme_id_t

```
typedef uint8_t hsm_signature_scheme_id_t
```

6.1.3.25 hsm_hash_algo_t

```
typedef uint8_t hsm_hash_algo_t
```

6.1.3.26 hsm_key_type_t

```
typedef uint8_t hsm_key_type_t
```

6.1.3.27 hsm_key_type_ext_t

```
typedef uint8_t hsm_key_type_ext_t
```

6.1.3.28 hsm_key_info_t

```
typedef uint16_t hsm_key_info_t
```

6.1.3.29 hsm_addr_msb_t

```
typedef uint32_t hsm_addr_msb_t
```

6.1.3.30 hsm_addr_lsb_t

```
typedef uint32_t hsm_addr_lsb_t
```

6.1.3.31 hsm_verification_status_t

```
typedef uint32_t hsm_verification_status_t
```

6.1.4 Enumeration Type Documentation**6.1.4.1 hsm_err_t**

```
enum hsm_err_t
```

Error codes returned by HSM functions.

Enumerator

HSM_NO_ERROR	Success.
HSM_INVALID_MESSAGE	The received message is invalid or unknown.
HSM_INVALID_ADDRESS	The provided address is invalid or doesn't respect the API requirements.
HSM_UNKNOWN_ID	The provided identifier is not known.
HSM_INVALID_PARAM	One of the parameter provided in the command is invalid.
HSM_NVM_ERROR	NVM generic issue.
HSM_OUT_OF_MEMORY	There is not enough memory to handle the requested operation.
HSM_UNKNOWN_HANDLE	Unknown session/service handle.
HSM_UNKNOWN_KEY_STORE	The key store identified by the provided "key store Id" doesn't exist and the "create" flag is not set.
HSM_KEY_STORE_AUTH	Key storage authentication fails.
HSM_KEY_STORAGE_ERROR	An error occurred in the key storage internal processing.
HSM_ID_CONFLICT	An element (key storage, key. . .) with the provided ID already exists.
HSM_RNG_NOT_STARTED	The internal RNG is not started.
HSM_CMD_NOT_SUPPORTED	The functionality is not supported for the current session/service/key store configuration.
HSM_INVALID_LIFECYCLE	Invalid lifecycle for requested operation.
HSM_KEY_STORE_CONFLICT	A key store with the same attributes already exists.
HSM_GENERAL_ERROR	Error not covered by other codes occurred.

6.1.5 Function Documentation

6.1.5.1 hsm_open_session()

```
hsm_err_t hsm_open_session (
    open_session_args_t * args,
    hsm_hdl_t * session_hdl )
```

Initiate a HSM session.

Parameters

<i>args</i>	pointer to the structure containing the function arguments.
<i>session_hdl</i>	pointer to where the session handle must be written.

Returns

`error_code` error code.

6.1.5.2 hsm_close_session()

```
hsm_err_t hsm_close_session (
    hsm_hdl_t session_hdl )
```

Terminate a previously opened HSM session

Parameters

<i>session_hdl</i>	pointer to the handle identifying the session to be closed.
--------------------	---

Returns

`error_code` error code.

6.1.5.3 hsm_open_key_store_service()

```
hsm_err_t hsm_open_key_store_service (
    hsm_hdl_t session_hdl,
    open_svc_key_store_args_t * args,
    hsm_hdl_t * key_store_hdl )
```

Open a service flow on the specified key store.

Parameters

<i>session_hdl</i>	pointer to the handle indentifying the current session.
<i>args</i>	pointer to the structure containing the function arugments.
<i>key_store_hdl</i>	pointer to where the key store service flow handle must be written.

Returns

`error_code` error code.

6.1.5.4 hsm_close_key_store_service()

```
hsm_err_t hsm_close_key_store_service (
    hsm_hdl_t key_store_hdl )
```

Close a previously opened key store service flow.

Parameters

<i>handle</i>	indentifying the key store service flow to be closed.
---------------	---

Returns

`error_code` error code.

6.1.5.5 hsm_open_key_management_service()

```
hsm_err_t hsm_open_key_management_service (
    hsm_hdl_t key_store_hdl,
    open_svc_key_management_args_t * args,
    hsm_hdl_t * key_management_hdl )
```

Open a key management service flow

User must open this service in order to perform operation on the key store content: key generate, delete, update

Parameters

<i>key_store_hdl</i>	handle indentifying the key store service flow.
<i>args</i>	pointer to the structure containing the function arugments.
<i>key_management_hdl</i>	pointer to where the key management service flow handle must be written.

Returns

error_code error code.

6.1.5.6 hsm_generate_key()

```
hsm_err_t hsm_generate_key (
    hsm_hdl_t key_management_hdl,
    op_generate_key_args_t args )
```

Generate a key or a key pair in the key store. In case of asymeric keys, the public key can optionally be exported. The generated key can be stored in a new or in an existing key slot with the restriction that an existing key can be replaced only by a key of the same type.

User can call this function only after having opened a key management service flow

Parameters

<i>key_management_hdl</i>	handle identifying the key management service flow.
<i>args</i>	pointer to the structure containing the function arugments.

Returns

error code

6.1.5.7 hsm_manage_key()

```
hsm_err_t hsm_manage_key (
    hsm_hdl_t key_management_hdl,
    op_manage_key_args_t * args )
```

This command is designed to perform operation on an existing key.

User can call this function only after having opened a key management service flow

Parameters

<i>key_management_hdl</i>	handle identifying the key management service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

6.1.5.8 hsm_butterfly_key_expansion()

```
hsm_err_t hsm_butterfly_key_expansion (
    hsm_hdl_t key_management_hdl,
    op_butt_key_exp_args_t * args )
```

This command is designed to perform the butterfly key expansion operation on an ECC private key in case of implicit certificate. Optionally the resulting public key is exported.

The result of the key expansion function is calculated outside the HSM and passed as input.

User can call this function only after having opened a key management service flow.

The following operation is performed:

$$\text{out_key} = (\text{Key} + \text{data1}) * \text{data2} + \text{data3} \pmod{n}$$

Explicit certificates: $\text{data1} = 0$, $\text{data2} = 1$ $\text{data3} = f1/f2(k, i, j)$,

$$\text{out_key} = \text{Key} + f1/f2(k, i, j) \pmod{n}$$

Implicit certificates: $\text{data1} = f1(k, i, j)$, $\text{data2} = \text{hash value used to in the derivation of the pseudonym ECC key}$

$\text{data3} = \text{private reconstruction value pij}$,

$$\text{out_key} = (\text{Key} + f1(k, i, j)) * \text{Hash} + \text{pij}$$

Parameters

<i>key_management_hdl</i>	handle identifying the key store management service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

6.1.5.9 hsm_close_key_management_service()

```
hsm_err_t hsm_close_key_management_service (
    hsm_hdl_t key_management_hdl )
```

Terminate a previously opened key management service flow

Parameters

<i>key_management_hdl</i>	handle identifying the key management service flow.
---------------------------	---

Returns

error code

6.1.5.10 hsm_open_cipher_service()

```
hsm_err_t hsm_open_cipher_service (
    hsm_hdl_t key_store_hdl,
    open_svc_cipher_args_t * args,
    hsm_hdl_t * chipper_hdl )
```

Open a cipher service flow

User can call this function only after having opened a key store service flow. User must open this service in order to perform cipher operations.

Parameters

<i>key_store_hdl</i>	handle indentifying the key store service flow.
<i>args</i>	pointer to the structure containing the function arugments.
<i>chipper_hdl</i>	pointer to where the cipher service flow handle must be written.

Returns

error code

6.1.5.11 hsm_cipher_one_go()

```
hsm_err_t hsm_cipher_one_go (
    hsm_hdl_t chipper_hdl,
    op_cipher_one_go_args_t * args )
```

Perform ciphering operation

User can call this function only after having opened a cipher service flow

Parameters

<i>chipper_hdl</i>	handle identifying the cipher service flow.
<i>args</i>	pointer to the structure containing the function arugments.

Returns

error code

6.1.5.12 hsm_ecies_decryption()

```
hsm_err_t hsm_ecies_decryption (
    hsm_hdl_t cipher_hdl,
    hsm_op_ecies_dec_args_t * args )
```

Decrypt data usign ECIES

User can call this function only after having opened a cipher store service flow

Parameters

<i>session_hdl</i>	handle identifying the current session.
<i>args</i>	pointer to the structure containing the function arugments.

Returns

error code

6.1.5.13 hsm_close_cipher_service()

```
hsm_err_t hsm_close_cipher_service (
    hsm_hdl_t chiper_hdl )
```

Terminate a previously opened cipher service flow

Parameters

<i>chiper_hdl</i>	pointer to handle identifying the cipher service flow to be closed.
-------------------	---

Returns

error code

6.1.5.14 hsm_open_signature_generation_service()

```
hsm_err_t hsm_open_signature_generation_service (
    hsm_hdl_t key_store_hdl,
    open_svc_sign_gen_args_t * args,
    hsm_hdl_t * signature_gen_hdl )
```

Open a signature generation service flow

User can call this function only after having opened a key store service flow. User must open this service in order to perform signature generation operations.

Parameters

<i>key_store_hdl</i>	handle indentifying the key store service flow.
<i>args</i>	pointer to the structure containing the function arugments.
<i>signature_gen_hdl</i>	pointer to where the signature generation service flow handle must be written.

Returns

error code

6.1.5.15 hsm_close_signature_generation_service()

```
hsm_err_t hsm_close_signature_generation_service (
    hsm_hdl_t signature_gen_hdl )
```

Terminate a previously opened signature generation service flow

Parameters

<i>signature_gen_hdl</i>	handle identifying the signature generation service flow to be closed.
--------------------------	--

Returns

error code

6.1.5.16 hsm_generate_signature()

```
hsm_err_t hsm_generate_signature (
    hsm_hdl_t signature_gen_hdl,
    op_generate_sign_args_t * args )
```

Generate a digital signature according to the signature scheme

User can call this function only after having opened a signature generation service flow The signature $S=(r,s)$ is always stored in format $r||s||R_y$ where R_y is an additional byte containing the lsb of y . The R_y validity is based on the “compressed point” flag.

Parameters

<i>signature_gen_hdl</i>	handle identifying the signature generation service flow
<i>args</i>	pointer to the structure containing the function arugments.

Returns

error code

6.1.5.17 hsm_prepare_signature()

```
hsm_err_t hsm_prepare_signature (
    hsm_hdl_t signature_gen_hdl,
    op_prepare_sign_args_t * args )
```

Prepare the creation of a signature by pre-calculating the operations having not dependencies on the input message. The pre-calculated value will be stored internally and used to the next call of hsm_generate_signature_finalize. User can call this function only after having opened a signature generation service flow. The signature $S=(r,s)$ is stored in format $r||s||R_y$ where R_y is an additional byte containing the lsb of y , the validity of the R_y parameter is based on the “compressed point” flag.

Parameters

<i>signature_gen_hdl</i>	handle identifying the signature generation service flow
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

6.1.5.18 hsm_finalize_signature()

```
hsm_err_t hsm_finalize_signature (
    hsm_hdl_t signature_gen_hdl,
    op_finalize_sign_args_t * args )
```

Finalize the computation of a digital signature. User can call this function only after having called the hsm_prepare_signature API.

Parameters

<i>signature_gen_hdl</i>	handle identifying the signature generation service flow
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

6.1.5.19 hsm_open_signature_verification_service()

```
hsm_err_t hsm_open_signature_verification_service (
    hsm_hdl_t session_hdl,
    open_svc_sign_ver_args_t * args,
    hsm_hdl_t * signature_ver_hdl )
```

User must open this service in order to perform signature verification operations. User can call this function only after having opened a session.

Parameters

<i>session_hdl</i>	handle indentifying the current session.
<i>args</i>	pointer to the structure containing the function arugments.
<i>signature_ver_hdl</i>	pointer to where the signature verification service flow handle must be written.

Returns

error code

6.1.5.20 hsm_verify_signature()

```
hsm_err_t hsm_verify_signature (
    hsm_hdl_t signature_ver_hdl,
    op_verify_sign_args_t * args,
    hsm_verification_status_t * status )
```

Verify a digital signature according to the signature scheme

User can call this function only after having opened a signature verification service flow. The signature $S=(r,s)$ is expected to be in format $r||s||R_y$ where R_y is an additional byte containing the lsb of y , the validity of the R_y parameters is based on the "compressed point" flag. Only not-compressed keys (x,y) can be used by this command. Compressed keys can be decompressed by using the dedicated API.

Parameters

<i>signature_ver_hdl</i>	handle identifying the signature verification service flow.
<i>args</i>	pointer to the structure containing the function arugments.
<i>status</i>	pointer to where the verification status must be stored if the verification suceed the value HSM_VERIFICATION_STATUS_SUCCESS is returned.

Returns

error code

6.1.5.21 hsm_import_public_key()

```
hsm_err_t hsm_import_public_key (
    hsm_hdl_t signature_ver_hdl,
    op_import_public_key_args_t * args,
    hsm_addr_lsb_t * int_key )
```

Import a public key to be used for several verification operations

User can call this function only after having opened a signature verification service flow. Only not-compressed keys (x,y) can be imprted by this command. Compressed keys can be decompressed by using the dedicated API.

Parameters

<i>signature_ver_hdl</i>	handle identifying the signature verification service flow.
<i>args</i>	pointer to the structure containing the function arugments.
<i>int_key</i>	pointer to where the key reference to be used as key in the hsm_verify_signature will be stored

Returns

error code

6.1.5.22 hsm_close_signature_verification_service()

```
hsm_err_t hsm_close_signature_verification_service (
    hsm_hdl_t signature_ver_hdl )
```

Terminate a previously opened signature verification service flow

Parameters

<i>signature_ver_hdl</i>	handle identifying the signature verification service flow to be closed.
--------------------------	--

Returns

error code

6.1.5.23 hsm_open_rng_service()

```
hsm_err_t hsm_open_rng_service (
    hsm_hdl_t session_hdl,
    open_svc_rng_args_t * args,
    hsm_hdl_t * rng_hdl )
```

Open a random number generation service flow

User can call this function only after having opened a session. User must open this service in order to perform rng operations.

Parameters

<i>session_hdl</i>	handle indentifying the current session.
<i>args</i>	pointer to the structure containing the function arugments.
<i>rng_hdl</i>	pointer to where the rng service flow handle must be written.

Returns

error code

6.1.5.24 hsm_close_rng_service()

```
hsm_err_t hsm_close_rng_service (
    hsm_hdl_t rng_hdl )
```

Terminate a previously opened rng service flow

Parameters

<i>rng_hdl</i>	handle identifying the rng service flow to be closed.
----------------	---

Returns

error code

6.1.5.25 hsm_get_random()

```
hsm_err_t hsm_get_random (
    hsm_hdl_t rng_hdl,
    op_get_random_args_t * args )
```

Get a freshly generated random number

User can call this function only after having opened a rng service flow

Parameters

<i>rng_hdl</i>	handle identifying the rng service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

6.1.5.26 hsm_open_hash_service()

```
hsm_err_t hsm_open_hash_service (
    hsm_hdl_t session_hdl,
    open_svc_hash_args_t * args,
    hsm_hdl_t * hash_hdl )
```

Open an hash service flow

User can call this function only after having opened a session. User must open this service in order to perform an hash operations.

Parameters

<i>session_hdl</i>	handle indentifying the current session.
<i>args</i>	pointer to the structure containing the function arguments.
<i>hash_hdl</i>	pointer to where the hash service flow handle must be written.

Returns

error code

6.1.5.27 hsm_close_hash_service()

```
hsm_err_t hsm_close_hash_service (
    hsm_hdl_t hash_hdl )
```

Terminate a previously opened hash service flow

Parameters

<i>hash_hdl</i>	handle identifying the hash service flow to be closed.
-----------------	--

Returns

error code

6.1.5.28 hsm_hash_one_go()

```
hsm_err_t hsm_hash_one_go (
    hsm_hdl_t hash_hdl,
    op_hash_one_go_args_t * args )
```

Perform the hash operation on a given input

User can call this function only after having opened a hash service flow

Parameters

<i>hash_hdl</i>	handle identifying the hash service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

6.1.5.29 hsm_pub_key_reconstruction()

```
hsm_err_t hsm_pub_key_reconstruction (
    hsm_hdl_t session_hdl,
    hsm_op_pub_key_rec_args_t * args )
```

Reconstruct an ECC public key provided by an implicit certificate

User can call this function only after having opened a session

This API implements the followign formula: $out_key = (pub_rec * hash) + ca_key$

Parameters

<i>session_hdl</i>	handle identifying the current session.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

6.1.5.30 hsm_pub_key_decompression()

```
hsm_err_t hsm_pub_key_decompression (
    hsm_hdl_t session_hdl,
    hsm_op_pub_key_dec_args_t * args )
```

6.1.5.31 hsm_ecies_encryption()

```
hsm_err_t hsm_ecies_encryption (
    hsm_hdl_t session_hdl,
    hsm_op_ecies_enc_args_t * args )
```

Encrypt data using ECIES

User can call this function only after having opened a session

Parameters

<i>session_hdl</i>	handle identifying the current session.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

7 Class Documentation

7.1 hsm_op_ecies_dec_args_t Struct Reference

Public Attributes

- `uint32_t key_identifier`
identifier of the private key to be used for the operation
- `hsm_addr_lsb_t input`
LSB of the address in the requester space where the input VCT can be found.
- `hsm_addr_lsb_t p1`

- LSB of the address in the requester space where the KDF P1 parameter can be found.*

 - [hsm_addr_lsb_t p2](#)

LSB of the address in the requester space where the MAC P2 parameter can be found.
 - [hsm_addr_lsb_t output](#)

LSB of the address in the requester space where the output plaintext must be written.
 - [uint32_t input_size](#)

length in bytes of the input VCT
 - [uint32_t output_size](#)

length in bytes of the output plaintext
 - [uint16_t p1_size](#)

length in bytes of the KDF P1 parameter
 - [uint16_t p2_size](#)

length in bytes of the MAC P2 parameter
 - [uint16_t mac_size](#)

length in bytes of the requested message authentication code
 - [hsm_key_type_t key_type](#)

indicates the type of the used key
 - [hsm_op_ecies_dec_flags_t flags](#)

bitmap specifying the operation attributes.

7.1.1 Member Data Documentation

7.1.1.1 key_identifier

```
uint32_t hsm_op_ecies_dec_args_t::key_identifier
```

identifier of the private key to be used for the operation

7.1.1.2 input

```
hsm\_addr\_lsb\_t hsm_op_ecies_dec_args_t::input
```

LSB of the address in the requester space where the input VCT can be found.

7.1.1.3 p1

```
hsm\_addr\_lsb\_t hsm_op_ecies_dec_args_t::p1
```

LSB of the address in the requester space where the KDF P1 parameter can be found.

7.1.1.4 p2

`hsm_addr_lsb_t hsm_op_ecies_dec_args_t::p2`

LSB of the address in the requester space where the MAC P2 parameter can be found.

7.1.1.5 output

`hsm_addr_lsb_t hsm_op_ecies_dec_args_t::output`

LSB of the address in the requester space where the output plaintext must be written.

7.1.1.6 input_size

`uint32_t hsm_op_ecies_dec_args_t::input_size`

length in bytes of the input VCT

7.1.1.7 output_size

`uint32_t hsm_op_ecies_dec_args_t::output_size`

length in bytes of the output plaintext

7.1.1.8 p1_size

`uint16_t hsm_op_ecies_dec_args_t::p1_size`

length in bytes of the KDF P1 parameter

7.1.1.9 p2_size

`uint16_t hsm_op_ecies_dec_args_t::p2_size`

length in bytes of the MAC P2 parameter

7.1.1.10 mac_size

`uint16_t hsm_op_ecies_dec_args_t::mac_size`

length in bytes of the requested message authentication code

7.1.1.11 key_type

[hsm_key_type_t](#) [hsm_op_ecies_dec_args_t::key_type](#)

indicates the type of the used key

7.1.1.12 flags

[hsm_op_ecies_dec_flags_t](#) [hsm_op_ecies_dec_args_t::flags](#)

bitmap specifying the operation attributes.

7.2 hsm_op_ecies_enc_args_t Struct Reference

Public Attributes

- [hsm_addr_msb_t](#) [input_ext](#)
MSB of the address in the requester space where the plaintext can be found.
- [hsm_addr_lsb_t](#) [input](#)
LSB of the address in the requester space where the plaintext can be found.
- [hsm_addr_msb_t](#) [pub_key_ext](#)
MSB of the address in the requester space where the recipient public key can be found.
- [hsm_addr_lsb_t](#) [pub_key](#)
LSB of the address in the requester space where the recipient public key can be found.
- [hsm_addr_msb_t](#) [p1_ext](#)
MSB of the address in the requester space where the KDF P1 parameter can be found.
- [hsm_addr_lsb_t](#) [p1](#)
LSB of the address in the requester space where the KDF P1 parameter can be found.
- [hsm_addr_msb_t](#) [p2_ext](#)
MSB of the address in the requester space where the MAC P2 parameter can be found.
- [hsm_addr_lsb_t](#) [p2](#)
LSB of the address in the requester space where the MAC P2 parameter can be found.
- [hsm_addr_msb_t](#) [output_ext](#)
MSB of the address in the requester space where the output VCT must be written.
- [hsm_addr_lsb_t](#) [output](#)
LSB of the address in the requester space where the output VCT must be written.
- [uint32_t](#) [input_size](#)
length in bytes of the input plaintext
- [uint16_t](#) [p1_size](#)
length in bytes of the KDF P1 parameter
- [uint16_t](#) [p2_size](#)
length in bytes of the MAC P2 parameter
- [uint16_t](#) [pub_key_size](#)
length in bytes of the recipient public key
- [uint16_t](#) [mac_size](#)
length in bytes of the requested message authentication code
- [uint32_t](#) [out_size](#)
length in bytes of the output VCT
- [hsm_key_type_t](#) [key_type](#)
indicates the type of the recipient public key
- [hsm_op_ecies_enc_flags_t](#) [flags](#)
bitmap specifying the operation attributes.
- [uint16_t](#) [rsv](#)

7.2.1 Member Data Documentation

7.2.1.1 input_ext

`hsm_addr_msb_t hsm_op_ecies_enc_args_t::input_ext`

MSB of the address in the requester space where the plaintext can be found.

7.2.1.2 input

`hsm_addr_lsb_t hsm_op_ecies_enc_args_t::input`

LSB of the address in the requester space where the plaintext can be found.

7.2.1.3 pub_key_ext

`hsm_addr_msb_t hsm_op_ecies_enc_args_t::pub_key_ext`

MSB of the address in the requester space where the recipient public key can be found.

7.2.1.4 pub_key

`hsm_addr_lsb_t hsm_op_ecies_enc_args_t::pub_key`

LSB of the address in the requester space where the recipient public key can be found.

7.2.1.5 p1_ext

`hsm_addr_msb_t hsm_op_ecies_enc_args_t::p1_ext`

MSB of the address in the requester space where the KDF P1 parameter can be found.

7.2.1.6 p1

`hsm_addr_lsb_t hsm_op_ecies_enc_args_t::p1`

LSB of the address in the requester space where the KDF P1 parameter can be found.

7.2.1.7 p2_ext

`hsm_addr_msb_t hsm_op_ecies_enc_args_t::p2_ext`

MSB of the address in the requester space where the MAC P2 parameter can be found.

7.2.1.8 p2

`hsm_addr_lsb_t hsm_op_ecies_enc_args_t::p2`

LSB of the address in the requester space where the MAC P2 parameter can be found.

7.2.1.9 output_ext

`hsm_addr_msb_t hsm_op_ecies_enc_args_t::output_ext`

MSB of the address in the requester space where the output VCT must be written.

7.2.1.10 output

`hsm_addr_lsb_t hsm_op_ecies_enc_args_t::output`

LSB of the address in the requester space where the output VCT must be written.

7.2.1.11 input_size

`uint32_t hsm_op_ecies_enc_args_t::input_size`

length in bytes of the input plaintext

7.2.1.12 p1_size

`uint16_t hsm_op_ecies_enc_args_t::p1_size`

length in bytes of the KDF P1 parameter

7.2.1.13 p2_size

`uint16_t hsm_op_ecies_enc_args_t::p2_size`

length in bytes of the MAC P2 parameter

7.2.1.14 pub_key_size

```
uint16_t hsm_op_ecies_enc_args_t::pub_key_size
```

length in bytes of the recipient public key

7.2.1.15 mac_size

```
uint16_t hsm_op_ecies_enc_args_t::mac_size
```

length in bytes of the requested message authentication code

7.2.1.16 out_size

```
uint32_t hsm_op_ecies_enc_args_t::out_size
```

length in bytes of the output VCT

7.2.1.17 key_type

```
hsm_key_type_t hsm_op_ecies_enc_args_t::key_type
```

indicates the type of the recipient public key

7.2.1.18 flags

```
hsm_op_ecies_enc_flags_t hsm_op_ecies_enc_args_t::flags
```

bitmap specifying the operation attributes.

7.2.1.19 rsv

```
uint16_t hsm_op_ecies_enc_args_t::rsv
```

7.3 hsm_op_pub_key_dec_args_t Struct Reference

Public Attributes

- [hsm_addr_msb_t key_ext](#)
MSB of the address in the requester space where the compressed ECC public key can be found. The expected key format is x||lsb_y where lsb_y is 1 byte having value 1 if the least-significant bit of the original (uncompressed) y coordinate is set, and 0 otherwise.
- [hsm_addr_lsb_t key](#)
LSB of the address in the requester space where the compressed ECC public key can be found. The expected key format is x||lsb_y where lsb_y is 1 byte having value 1 if the least-significant bit of the original (uncompressed) y coordinate is set, and 0 otherwise.
- [hsm_addr_msb_t out_key_ext](#)
MSB of the address in the requester space where the output resulting key must be written.
- [hsm_addr_lsb_t out_key](#)
LSB of the address in the requester space where the output resulting key must be written.
- [uint16_t key_size](#)
length in bytes of the input compressed public key
- [uint16_t out_key_size](#)
length in bytes of the resulting public key
- [hsm_key_type_t key_type](#)
indicates the type of the manged keys.
- [hsm_op_pub_key_dec_flags_t flags](#)
bitmap specifying the operation attributes.
- [uint16_t rsv](#)

7.3.1 Detailed Description

Decompress an ECC public key

The expected key format is x||lsb_y where lsb_y is 1 byte having value 1 if the least-significant bit of the original (uncompressed) y coordinate is set, and 0 otherwise. User can call this function only after having opened a session

Parameters

<i>session_hdl</i>	handle identifying the current session.
<i>args</i>	pointer to the structure containing the function arugments.

Returns

error code

7.3.2 Member Data Documentation

7.3.2.1 key_ext

[hsm_addr_msb_t](#) hsm_op_pub_key_dec_args_t::key_ext

MSB of the address in the requester space where the compressed ECC public key can be found. The expected key format is x||lsb_y where lsb_y is 1 byte having value 1 if the least-significant bit of the original (uncompressed) y coordinate is set, and 0 otherwise.

7.3.2.2 key

`hsm_addr_lsb_t hsm_op_pub_key_dec_args_t::key`

LSB of the address in the requester space where the compressed ECC public key can be found. The expected key format is `x||lsb_y` where `lsb_y` is 1 byte having value 1 if the least-significant bit of the original (uncompressed) `y` coordinate is set, and 0 otherwise.

7.3.2.3 out_key_ext

`hsm_addr_msb_t hsm_op_pub_key_dec_args_t::out_key_ext`

MSB of the address in the requester space where the output resulting key must be written.

7.3.2.4 out_key

`hsm_addr_lsb_t hsm_op_pub_key_dec_args_t::out_key`

LSB of the address in the requester space where the output resulting key must be written.

7.3.2.5 key_size

`uint16_t hsm_op_pub_key_dec_args_t::key_size`

length in bytes of the input compressed public key

7.3.2.6 out_key_size

`uint16_t hsm_op_pub_key_dec_args_t::out_key_size`

length in bytes of the resulting public key

7.3.2.7 key_type

`hsm_key_type_t hsm_op_pub_key_dec_args_t::key_type`

indicates the type of the manged keys.

7.3.2.8 flags

[hsm_op_pub_key_dec_flags_t](#) `hsm_op_pub_key_dec_args_t::flags`

bitmap specifying the operation attributes.

7.3.2.9 rsv

`uint16_t hsm_op_pub_key_dec_args_t::rsv`

7.4 hsm_op_pub_key_rec_args_t Struct Reference

Public Attributes

- [hsm_addr_msb_t pub_rec_ext](#)
MSB of the address in the requester space where the public reconstruction value extracted from the implicit certificate can be found.
- [hsm_addr_msb_t pub_rec](#)
LSB of the address in the requester space where the public reconstruction value extracted from the implicit certificate can be found.
- [hsm_addr_msb_t hash_ext](#)
MSB of the address in the requester space where the hash value can be found. In the butterfly scheme it corresponds to the hash value calculated over PCA certificate and, concatenated, the implicit certificat.
- [hsm_addr_lsb_t hash](#)
LSB of the address in the requester space where the hash value can be found. In the butterfly scheme it corresponds to the hash value calculated over PCA certificate and, concatenated, the implicit certificat.
- [hsm_addr_msb_t ca_key_ext](#)
MSB of the address in the requester space where the CA public key can be found.
- [hsm_addr_lsb_t ca_key](#)
LSB of the address in the requester space where the CA public key can be found.
- [hsm_addr_msb_t out_key_ext](#)
MSB of the address in the requester space where the output resulting key must be written.
- [hsm_addr_lsb_t out_key](#)
LSB of the address in the requester space where the output resulting key must be written.
- `uint16_t pub_rec_size`
length in bytes of the public reconstruction value
- `uint16_t hash_size`
length in bytes of the input hash
- `uint16_t ca_key_size`
length in bytes of the input CA public key
- `uint16_t out_key_size`
length in bytes of the output key
- [hsm_key_type_t key_type](#)
indicates the type of the manged keys.
- [hsm_op_pub_key_rec_flags_t flags](#)
flags bitmap specifying the operation attributes.
- `uint16_t rsv`

7.4.1 Member Data Documentation

7.4.1.1 pub_rec_ext

`hsm_addr_msb_t hsm_op_pub_key_rec_args_t::pub_rec_ext`

MSB of the address in the requester space where the public reconstruction value extracted from the implicit certificate can be found.

7.4.1.2 pub_rec

`hsm_addr_msb_t hsm_op_pub_key_rec_args_t::pub_rec`

LSB of the address in the requester space where the public reconstruction value extracted from the implicit certificate can be found.

7.4.1.3 hash_ext

`hsm_addr_msb_t hsm_op_pub_key_rec_args_t::hash_ext`

MSB of the address in the requester space where the hash value can be found. In the butterfly scheme it corresponds to the hash value calculated over PCA certificate and, concatenated, the implicit certificate.

7.4.1.4 hash

`hsm_addr_lsb_t hsm_op_pub_key_rec_args_t::hash`

LSB of the address in the requester space where the hash value can be found. In the butterfly scheme it corresponds to the hash value calculated over PCA certificate and, concatenated, the implicit certificate.

7.4.1.5 ca_key_ext

`hsm_addr_msb_t hsm_op_pub_key_rec_args_t::ca_key_ext`

MSB of the address in the requester space where the CA public key can be found.

7.4.1.6 ca_key

`hsm_addr_lsb_t hsm_op_pub_key_rec_args_t::ca_key`

LSB of the address in the requester space where the CA public key can be found.

7.4.1.7 out_key_ext

`hsm_addr_msb_t` `hsm_op_pub_key_rec_args_t::out_key_ext`

MSB of the address in the requester space where the output resulting key must be written.

7.4.1.8 out_key

`hsm_addr_lsb_t` `hsm_op_pub_key_rec_args_t::out_key`

LSB of the address in the requester space where the output resulting key must be written.

7.4.1.9 pub_rec_size

`uint16_t` `hsm_op_pub_key_rec_args_t::pub_rec_size`

length in bytes of the public reconstruction value

7.4.1.10 hash_size

`uint16_t` `hsm_op_pub_key_rec_args_t::hash_size`

length in bytes of the input hash

7.4.1.11 ca_key_size

`uint16_t` `hsm_op_pub_key_rec_args_t::ca_key_size`

length in bytes of the input CA public key

7.4.1.12 out_key_size

`uint16_t` `hsm_op_pub_key_rec_args_t::out_key_size`

length in bytes of the output key

7.4.1.13 key_type

`hsm_key_type_t` `hsm_op_pub_key_rec_args_t::key_type`

indicates the type of the manged keys.

7.4.1.14 flags

`hsm_op_pub_key_rec_flags_t` `hsm_op_pub_key_rec_args_t::flags`

flags bitmap specifying the operation attributes.

7.4.1.15 rsv

`uint16_t` `hsm_op_pub_key_rec_args_t::rsv`

7.5 op_but_key_exp_args_t Struct Reference

Public Attributes

- `uint32_t` `key_identifier`
identifier of the key to be expanded
- `hsm_addr_lsb_t` `data1`
LSB of the address in the requester space where the data1 input can be found.
- `hsm_addr_lsb_t` `data2`
LSB of the address in the requester space where the data2 input can be found.
- `hsm_addr_lsb_t` `data3`
LSB of the address in the requester space where the data3 input can be found.
- `uint8_t` `data1_size`
length in bytes of the add_data1 input
- `uint8_t` `data2_size`
length in bytes of the add_data2 input
- `uint8_t` `data3_size`
length in bytes of the data3 input
- `hsm_op_but_key_exp_flags_t` `flags`
bitmap specifying the operation properties
- `uint32_t` `dest_key_identifier`
identifier of the derived key
- `hsm_addr_lsb_t` `output`
LSB of the address in the requester space where the public key must be written.
- `uint16_t` `output_size`
length in bytes of the output area, if the size is 0, no key is copied in the output.
- `hsm_key_type_t` `key_type`
indicates the type of the key to be managed.
- `uint8_t` `rsv`

7.5.1 Member Data Documentation

7.5.1.1 key_identifier

`uint32_t op_butt_key_exp_args_t::key_identifier`

identifier of the key to be expanded

7.5.1.2 data1

`hsm_addr_lsb_t op_butt_key_exp_args_t::data1`

LSB of the address in the requester space where the data1 input can be found.

7.5.1.3 data2

`hsm_addr_lsb_t op_butt_key_exp_args_t::data2`

LSB of the address in the requester space where the data2 input can be found.

7.5.1.4 data3

`hsm_addr_lsb_t op_butt_key_exp_args_t::data3`

LSB of the address in the requester space where the data3 input can be found.

7.5.1.5 data1_size

`uint8_t op_butt_key_exp_args_t::data1_size`

length in bytes of the add_data1 input

7.5.1.6 data2_size

`uint8_t op_butt_key_exp_args_t::data2_size`

length in bytes of the add_data2 input

7.5.1.7 data3_size

`uint8_t op_butt_key_exp_args_t::data3_size`

length in bytes of the data3 input

7.5.1.8 flags

`hsm_op_but_key_exp_flags_t` `op_but_key_exp_args_t::flags`

bitmap specifying the operation properties

7.5.1.9 dest_key_identifier

`uint32_t` `op_but_key_exp_args_t::dest_key_identifier`

identifier of the derived key

7.5.1.10 output

`hsm_addr_lsb_t` `op_but_key_exp_args_t::output`

LSB of the address in the requester space where the public key must be written.

7.5.1.11 output_size

`uint16_t` `op_but_key_exp_args_t::output_size`

length in bytes of the output area, if the size is 0, no key is copied in the output.

7.5.1.12 key_type

`hsm_key_type_t` `op_but_key_exp_args_t::key_type`

indicates the type of the key to be managed.

7.5.1.13 rsv

`uint8_t` `op_but_key_exp_args_t::rsv`

7.6 op_cipher_one_go_args_t Struct Reference

Public Attributes

- [uint32_t key_identifier](#)
identifier of the key to be used for the operation
- [hsm_addr_lsb_t iv](#)
LSB of the address in the requester space where the initialization vector can be found.
- [uint16_t iv_size](#)
*length in bytes of the initialization vector
it must be 0 for algorithms not using the initialization vector.
It must be 12 for AES in CCM mode*
- [hsm_op_cipher_one_go_algo_t cipher_algo](#)
algorithm to be used for the operation
- [hsm_op_cipher_one_go_flags_t flags](#)
bitmap specifying the operation attributes
- [hsm_addr_lsb_t input](#)
*LSB of the address in the requester space where the input to be processed can be found
plaintext for encryption
ciphertext for decryption (tag is concatenated for CCM)*
- [hsm_addr_lsb_t output](#)
*LSB of the address in the requester space where the output must be stored
ciphertext for encryption (tag is concatenated for CCM)
plaintext for decryption.*
- [uint32_t input_size](#)
length in bytes of the input
- [uint32_t output_size](#)
length in bytes of the output

7.6.1 Member Data Documentation

7.6.1.1 key_identifier

```
uint32_t op_cipher_one_go_args_t::key_identifier
```

identifier of the key to be used for the operation

7.6.1.2 iv

```
hsm_addr_lsb_t op_cipher_one_go_args_t::iv
```

LSB of the address in the requester space where the initialization vector can be found.

7.6.1.3 iv_size

```
uint16_t op_cipher_one_go_args_t::iv_size
```

length in bytes of the initialization vector
it must be 0 for algorithms not using the initialization vector.
It must be 12 for AES in CCM mode

7.6.1.4 cipher_algo

```
hsm_op_cipher_one_go_algo_t op_cipher_one_go_args_t::cipher_algo
```

algorithm to be used for the operation

7.6.1.5 flags

```
hsm_op_cipher_one_go_flags_t op_cipher_one_go_args_t::flags
```

bitmap specifying the operation attributes

7.6.1.6 input

```
hsm_addr_lsb_t op_cipher_one_go_args_t::input
```

LSB of the address in the requester space where the input to be processed can be found
plaintext for encryption
ciphertext for decryption (tag is concatenated for CCM)

7.6.1.7 output

```
hsm_addr_lsb_t op_cipher_one_go_args_t::output
```

LSB of the address in the requester space where the output must be stored
ciphertext for encryption (tag is concatenated for CCM)
plaintext for decryption.

7.6.1.8 input_size

```
uint32_t op_cipher_one_go_args_t::input_size
```

length in bytes of the input

7.6.1.9 output_size

```
uint32_t op_cipher_one_go_args_t::output_size
```

length in bytes of the output

7.7 op_finalize_sign_args_t Struct Reference

Public Attributes

- [uint32_t key_identifier](#)
identifier of the key to be used for the operation
- [hsm_addr_lsb_t message](#)
LSB of the address in the requester space where the input (message or message digest) to be processed can be found.
- [hsm_addr_lsb_t signature](#)
LSB of the address in the requester space where the signature must be stored. The signature $S=(r,s)$ is stored in format $r||s||R_y$ where R_y is an additional byte containing the lsb of y , the validity of the R_y parameter is based on the "compressed point" flag.
- [uint32_t message_size](#)
length in bytes of the input
- [uint16_t signature_size](#)
length in bytes of the output
- [hsm_op_finalize_sign_flags_t flags](#)
bitmap specifying the operation attributes
- [uint8_t rsv](#)

7.7.1 Member Data Documentation

7.7.1.1 key_identifier

```
uint32_t op_finalize_sign_args_t::key_identifier
```

identifier of the key to be used for the operation

7.7.1.2 message

```
hsm\_addr\_lsb\_t op_finalize_sign_args_t::message
```

LSB of the address in the requester space where the input (message or message digest) to be processed can be found.

7.7.1.3 signature

`hsm_addr_lsb_t op_finalize_sign_args_t::signature`

LSB of the address in the requester space where the signature must be stored. The signature $S=(r,s)$ is stored in format $r||s||R_y$ where R_y is an additional byte containing the lsb of y , the validity of the R_y parameter is based on the “compressed point” flag.

7.7.1.4 message_size

`uint32_t op_finalize_sign_args_t::message_size`

length in bytes of the input

7.7.1.5 signature_size

`uint16_t op_finalize_sign_args_t::signature_size`

length in bytes of the output

7.7.1.6 flags

`hsm_op_finalize_sign_flags_t op_finalize_sign_args_t::flags`

bitmap specifying the operation attributes

7.7.1.7 rsv

`uint8_t op_finalize_sign_args_t::rsv`

7.8 op_generate_key_args_t Struct Reference

Public Attributes

- `uint32_t * key_identifier`
pointer to the identifier of the key to be used for the operation.
In case of create operation the new key identifier will be stored in this location.
- `uint16_t out_size`
length in bytes of the output area, if the size is 0, no key is copied in the output.
- `hsm_op_key_gen_flags_t flags`
bitmap specifying the operation properties.
- `uint8_t rsv`
- `hsm_key_type_t key_type`
indicates which type of key must be generated.
- `hsm_key_type_ext_t key_type_ext`
- `hsm_key_info_t key_info`
bitmap specifying the properties of the key.
- `hsm_addr_lsb_t out_key`
LSB of the address in the requester space where to store the public key.

7.8.1 Member Data Documentation

7.8.1.1 key_identifier

`uint32_t* op_generate_key_args_t::key_identifier`

pointer to the identifier of the key to be used for the operation.
In case of create operation the new key identifier will be stored in this location.

7.8.1.2 out_size

`uint16_t op_generate_key_args_t::out_size`

length in bytes of the output area, if the size is 0, no key is copied in the output.

7.8.1.3 flags

`hsm_op_key_gen_flags_t op_generate_key_args_t::flags`

bitmap specifying the operation properties.

7.8.1.4 rsv

`uint8_t op_generate_key_args_t::rsv`

7.8.1.5 key_type

`hsm_key_type_t op_generate_key_args_t::key_type`

indicates which type of key must be generated.

7.8.1.6 key_type_ext

`hsm_key_type_ext_t op_generate_key_args_t::key_type_ext`

7.8.1.7 key_info

`hsm_key_info_t op_generate_key_args_t::key_info`

bitmap specifying the properties of the key.

7.8.1.8 out_key

`hsm_addr_lsb_t op_generate_key_args_t::out_key`

LSB of the address in the requester space where to store the public key.

7.9 op_generate_sign_args_t Struct Reference

Public Attributes

- `uint32_t key_identifier`
identifier of the key to be used for the operation
- `hsm_addr_lsb_t message`
LSB of the address in the requester space where the input (message or message digest) to be processed can be found.
- `hsm_addr_lsb_t signature`
LSB of the address in the requester space where the signature must be stored. The signature $S=(r,s)$ is always stored in format $r||s||Ry$ where Ry is an additional byte containing the lsb of y . The Ry validity is based on the “compressed point” flag.
- `uint32_t message_size`
length in bytes of the input
- `uint16_t signature_size`
length in bytes of the output
- `hsm_signature_scheme_id_t scheme_id`
identifier of the digital signature scheme to be used for the operation
- `hsm_op_generate_sign_flags_t flags`
bitmap specifying the operation attributes

7.9.1 Member Data Documentation

7.9.1.1 key_identifier

`uint32_t op_generate_sign_args_t::key_identifier`

identifier of the key to be used for the operation

7.9.1.2 message

`hsm_addr_lsb_t op_generate_sign_args_t::message`

LSB of the address in the requester space where the input (message or message digest) to be processed can be found.

7.9.1.3 signature

`hsm_addr_lsb_t op_generate_sign_args_t::signature`

LSB of the address in the requester space where the signature must be stored. The signature $S=(r,s)$ is always stored in format $r||s||R_y$ where R_y is an additional byte containing the lsb of y . The R_y validity is based on the "compressed point" flag.

7.9.1.4 message_size

`uint32_t op_generate_sign_args_t::message_size`

length in bytes of the input

7.9.1.5 signature_size

`uint16_t op_generate_sign_args_t::signature_size`

length in bytes of the output

7.9.1.6 scheme_id

`hsm_signature_scheme_id_t op_generate_sign_args_t::scheme_id`

identifier of the digital signature scheme to be used for the operation

7.9.1.7 flags

`hsm_op_generate_sign_flags_t op_generate_sign_args_t::flags`

bitmap specifying the operation attributes

7.10 `op_get_random_args_t` Struct Reference

Public Attributes

- [hsm_addr_lsb_t](#) `output`
LSB of the address in the requester space where the out random number must be written.
- `uint32_t` [random_size](#)
length in bytes of the random number to be provided.

7.10.1 Member Data Documentation

7.10.1.1 `output`

[hsm_addr_lsb_t](#) `op_get_random_args_t::output`

LSB of the address in the requester space where the out random number must be written.

7.10.1.2 `random_size`

`uint32_t` `op_get_random_args_t::random_size`

length in bytes of the random number to be provided.

7.11 `op_hash_one_go_args_t` Struct Reference

Public Attributes

- [hsm_addr_lsb_t](#) `input`
LSB of the address in the requester space where the input payload can be found.
- [hsm_addr_lsb_t](#) `output`
LSB of the address in the requester space where the output digest must be written.
- `uint32_t` [input_size](#)
length in bytes of the input
- `uint32_t` [output_size](#)
length in bytes of the output
- [hsm_hash_algo_t](#) `algo`
hash algorithm to be used for the operation
- [hsm_op_hash_one_go_flags_t](#) `flags`
flags bitmap specifying the operation attributes.
- `uint16_t` [rsv](#)

7.11.1 Member Data Documentation

7.11.1.1 input

`hsm_addr_lsb_t op_hash_one_go_args_t::input`

LSB of the address in the requester space where the input payload can be found.

7.11.1.2 output

`hsm_addr_lsb_t op_hash_one_go_args_t::output`

LSB of the address in the requester space where the output digest must be written.

7.11.1.3 input_size

`uint32_t op_hash_one_go_args_t::input_size`

length in bytes of the input

7.11.1.4 output_size

`uint32_t op_hash_one_go_args_t::output_size`

length in bytes of the output

7.11.1.5 algo

`hsm_hash_algo_t op_hash_one_go_args_t::algo`

hash algorithm to be used for the operation

7.11.1.6 flags

`hsm_op_hash_one_go_flags_t op_hash_one_go_args_t::flags`

flags bitmap specifying the operation attributes.

7.11.1.7 rsv

`uint16_t op_hash_one_go_args_t::rsv`

7.12 op_import_public_key_args_t Struct Reference

Public Attributes

- [hsm_addr_lsb_t](#) *key*
LSB of the address in the requester space where the public key to be imported can be found.
- [uint16_t](#) *key_size*
length in bytes of the input key
- [hsm_key_type_t](#) *key_type*
indicates the type of the key to be imported.
- [hsm_op_verify_sign_flags_t](#) *flags*
bitmap specifying the operation attributes

7.12.1 Member Data Documentation

7.12.1.1 key

[hsm_addr_lsb_t](#) op_import_public_key_args_t::key

LSB of the address in the requester space where the public key to be imported can be found.

7.12.1.2 key_size

[uint16_t](#) op_import_public_key_args_t::key_size

length in bytes of the input key

7.12.1.3 key_type

[hsm_key_type_t](#) op_import_public_key_args_t::key_type

indicates the type of the key to be imported.

7.12.1.4 flags

[hsm_op_verify_sign_flags_t](#) op_import_public_key_args_t::flags

bitmap specifying the operation attributes

7.13 op_manage_key_args_t Struct Reference

Public Attributes

- `uint32_t * key_identifier`
*pointer to the identifier of the key to be used for the operation.
In case of create operation the new key identifier will be stored in this location.*
- `uint16_t input_size`
length in bytes of the input key area. Not checked in case of delete operation.
- `hsm_op_manage_key_flags_t flags`
bitmap specifying the operation properties.
- `uint16_t rsv`
- `hsm_key_type_t key_type`
indicates the type of the key to be managed.
- `hsm_key_type_ext_t key_type_ext`
- `hsm_key_info_t key_info`
bitmap specifying the properties of the key, it will replace the existing value. Not checked in case of delete operation.
- `hsm_addr_lsb_t input_key`
LSB of the address in the requester space where the new key value can be found. Not checked in case of delete operation.

7.13.1 Member Data Documentation

7.13.1.1 key_identifier

```
uint32_t* op_manage_key_args_t::key_identifier
```

pointer to the identifier of the key to be used for the operation.
In case of create operation the new key identifier will be stored in this location.

7.13.1.2 input_size

```
uint16_t op_manage_key_args_t::input_size
```

length in bytes of the input key area. Not checked in case of delete operation.

7.13.1.3 flags

```
hsm_op_manage_key_flags_t op_manage_key_args_t::flags
```

bitmap specifying the operation properties.

7.13.1.4 rsv

`uint16_t op_manage_key_args_t::rsv`

7.13.1.5 key_type

`hsm_key_type_t op_manage_key_args_t::key_type`

indicates the type of the key to be managed.

7.13.1.6 key_type_ext

`hsm_key_type_ext_t op_manage_key_args_t::key_type_ext`

7.13.1.7 key_info

`hsm_key_info_t op_manage_key_args_t::key_info`

bitmap specifying the properties of the key, it will replace the existing value. Not checked in case of delete operation.

7.13.1.8 input_key

`hsm_addr_lsb_t op_manage_key_args_t::input_key`

LSB of the address in the requester space where the new key value can be found. Not checked in case of delete operation.

7.14 op_prepare_sign_args_t Struct Reference

Public Attributes

- `hsm_signature_scheme_id_t scheme_id`
identifier of the digital signature scheme to be used for the operation
- `hsm_op_prepare_signature_flags_t flags`
bitmap specifying the operation attributes
- `uint16_t rsv`

7.14.1 Member Data Documentation

7.14.1.1 scheme_id

`hsm_signature_scheme_id_t op_prepare_sign_args_t::scheme_id`

identifier of the digital signature scheme to be used for the operation

7.14.1.2 flags

`hsm_op_prepare_signature_flags_t op_prepare_sign_args_t::flags`

bitmap specifying the operation attributes

7.14.1.3 rsv

`uint16_t op_prepare_sign_args_t::rsv`

7.15 op_verify_sign_args_t Struct Reference

Public Attributes

- `hsm_addr_lsb_t key`
LSB of the address in the requester space where the public key to be used for the verification can be found.
- `hsm_addr_lsb_t message`
LSB of the address in the requester space where the input (message or message digest) to be processed can be found.
- `hsm_addr_lsb_t signature`
LSB of the address in the requester space where the signature can be found. The signature $S=(r,s)$ is expected to be in format $r||s||R_y$ where R_y is an additional byte containing the lsb of y , the validity of the R_y parameter is based on the "compressed point" flag.
- `uint16_t key_size`
length in bytes of the input key
- `uint16_t signature_size`
length in bytes of the output - it must contains one additional byte where to store the R_y .
- `uint32_t message_size`
length in bytes of the input message
- `hsm_signature_scheme_id_t scheme_id`
identifier of the digital signature scheme to be used for the operation
- `hsm_op_verify_sign_flags_t flags`
bitmap specifying the operation attributes
- `uint16_t rsv`

7.15.1 Member Data Documentation

7.15.1.1 key

```
hsm_addr_lsb_t op_verify_sign_args_t::key
```

LSB of the address in the requester space where the public key to be used for the verification can be found.

7.15.1.2 message

```
hsm_addr_lsb_t op_verify_sign_args_t::message
```

LSB of the address in the requester space where the input (message or message digest) to be processed can be found.

7.15.1.3 signature

```
hsm_addr_lsb_t op_verify_sign_args_t::signature
```

LSB of the address in the requester space where the signature can be found. The signature $S=(r,s)$ is expected to be in format $r||s||R_y$ where R_y is an additional byte containing the lsb of y , the validity of the R_y parameter is based on the “compressed point” flag.

7.15.1.4 key_size

```
uint16_t op_verify_sign_args_t::key_size
```

length in bytes of the input key

7.15.1.5 signature_size

```
uint16_t op_verify_sign_args_t::signature_size
```

length in bytes of the output - it must contains one additional byte where to store the R_y .

7.15.1.6 message_size

```
uint32_t op_verify_sign_args_t::message_size
```

length in bytes of the input message

7.15.1.7 scheme_id

```
hsm_signature_scheme_id_t op_verify_sign_args_t::scheme_id
```

identifier of the digital signature scheme to be used for the operation

7.15.1.8 flags

```
hsm_op_verify_sign_flags_t op_verify_sign_args_t::flags
```

bitmap specifying the operation attributes

7.15.1.9 rsv

```
uint16_t op_verify_sign_args_t::rsv
```

7.16 open_session_args_t Struct Reference

Public Attributes

- `uint8_t session_priority`
*not supported in current release, any value accepted. */*
- `uint8_t operating_mode`
*not supported in current release, any value accepted. */*
- `uint16_t rsv`

7.16.1 Member Data Documentation

7.16.1.1 session_priority

```
uint8_t open_session_args_t::session_priority
```

not supported in current release, any value accepted. */

7.16.1.2 operating_mode

```
uint8_t open_session_args_t::operating_mode
```

not supported in current release, any value accepted. */

7.16.1.3 rsv

`uint16_t open_session_args_t::rsv`

7.17 open_svc_cipher_args_t Struct Reference

Public Attributes

- [hsm_addr_msb_t input_address_ext](#)
most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_addr_msb_t output_address_ext](#)
most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_svc_cipher_flags_t flags](#)
bitmap specifying the services properties.
- `uint8_t rsv [3]`

7.17.1 Member Data Documentation

7.17.1.1 input_address_ext

`hsm_addr_msb_t open_svc_cipher_args_t::input_address_ext`

most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.

7.17.1.2 output_address_ext

`hsm_addr_msb_t open_svc_cipher_args_t::output_address_ext`

most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.

7.17.1.3 flags

`hsm_svc_cipher_flags_t open_svc_cipher_args_t::flags`

bitmap specifying the services properties.

7.17.1.4 rsv

`uint8_t open_svc_cipher_args_t::rsv[3]`

7.18 open_svc_hash_args_t Struct Reference

Public Attributes

- [hsm_addr_msb_t](#) `input_address_ext`
most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_addr_msb_t](#) `output_address_ext`
most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_svc_hash_flags_t](#) `flags`
bitmap indicating the service flow properties
- `uint8_t` `rsv` [3]

7.18.1 Member Data Documentation

7.18.1.1 input_address_ext

[hsm_addr_msb_t](#) open_svc_hash_args_t::input_address_ext

most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.

7.18.1.2 output_address_ext

[hsm_addr_msb_t](#) open_svc_hash_args_t::output_address_ext

most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.

7.18.1.3 flags

[hsm_svc_hash_flags_t](#) open_svc_hash_args_t::flags

bitmap indicating the service flow properties

7.18.1.4 rsv

`uint8_t` open_svc_hash_args_t::rsv[3]

7.19 open_svc_key_management_args_t Struct Reference

Public Attributes

- [hsm_addr_msb_t input_address_ext](#)
most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_addr_msb_t output_address_ext](#)
most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_svc_key_management_flags_t flags](#)
bitmap specifying the services properties.
- `uint8_t rsv [3]`

7.19.1 Member Data Documentation

7.19.1.1 input_address_ext

[hsm_addr_msb_t](#) open_svc_key_management_args_t::input_address_ext

most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.

7.19.1.2 output_address_ext

[hsm_addr_msb_t](#) open_svc_key_management_args_t::output_address_ext

most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.

7.19.1.3 flags

[hsm_svc_key_management_flags_t](#) open_svc_key_management_args_t::flags

bitmap specifying the services properties.

7.19.1.4 rsv

`uint8_t` open_svc_key_management_args_t::rsv[3]

7.20 open_svc_key_store_args_t Struct Reference

Public Attributes

- uint32_t [key_store_identifier](#)
user defined id identifying the key store./*
- uint32_t [authentication_nonce](#)
*user defined nonce used as authentication proof for accesing the key storage. */*
- uint16_t [max_updates_number](#)
*maximum number of updates authorized for the storage. Valid only for create operation. */*
- [hsm_svc_key_store_flags_t](#) flags
*bitmap specifying the services properties. */*
- uint8_t rsv

7.20.1 Member Data Documentation

7.20.1.1 key_store_identifier

uint32_t open_svc_key_store_args_t::key_store_identifier

user defined id identifying the key store.*/

7.20.1.2 authentication_nonce

uint32_t open_svc_key_store_args_t::authentication_nonce

user defined nonce used as authentication proof for accesing the key storage. */

7.20.1.3 max_updates_number

uint16_t open_svc_key_store_args_t::max_updates_number

maximum number of updates authorized for the storage. Valid only for create operation. */

7.20.1.4 flags

[hsm_svc_key_store_flags_t](#) open_svc_key_store_args_t::flags

bitmap specifying the services properties. */

7.20.1.5 rsv

`uint8_t open_svc_key_store_args_t::rsv`

7.21 open_svc_rng_args_t Struct Reference

Public Attributes

- [hsm_addr_msb_t input_address_ext](#)
most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_addr_msb_t output_address_ext](#)
most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_svc_rng_flags_t flags](#)
bitmap indicating the service flow properties
- `uint8_t rsv` [3]

7.21.1 Member Data Documentation

7.21.1.1 input_address_ext

[hsm_addr_msb_t](#) `open_svc_rng_args_t::input_address_ext`

most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.

7.21.1.2 output_address_ext

[hsm_addr_msb_t](#) `open_svc_rng_args_t::output_address_ext`

most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.

7.21.1.3 flags

[hsm_svc_rng_flags_t](#) `open_svc_rng_args_t::flags`

bitmap indicating the service flow properties

7.21.1.4 rsv

`uint8_t open_svc_rng_args_t::rsv` [3]

7.22 open_svc_sign_gen_args_t Struct Reference

Public Attributes

- [hsm_addr_msb_t input_address_ext](#)
most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_addr_msb_t output_address_ext](#)
most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_svc_signature_generation_flags_t flags](#)
bitmap specifying the services properties.
- `uint8_t rsv [3]`

7.22.1 Member Data Documentation

7.22.1.1 input_address_ext

[hsm_addr_msb_t](#) open_svc_sign_gen_args_t::input_address_ext

most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.

7.22.1.2 output_address_ext

[hsm_addr_msb_t](#) open_svc_sign_gen_args_t::output_address_ext

most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.

7.22.1.3 flags

[hsm_svc_signature_generation_flags_t](#) open_svc_sign_gen_args_t::flags

bitmap specifying the services properties.

7.22.1.4 rsv

`uint8_t` open_svc_sign_gen_args_t::rsv[3]

7.23 open_svc_sign_ver_args_t Struct Reference

Public Attributes

- [hsm_addr_msb_t input_address_ext](#)
most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_addr_msb_t output_address_ext](#)
most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.
- [hsm_svc_signature_verification_flags_t flags](#)
bitmap indicating the service flow properties
- `uint8_t rsv [3]`

7.23.1 Member Data Documentation

7.23.1.1 input_address_ext

`hsm_addr_msb_t open_svc_sign_ver_args_t::input_address_ext`

most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the commands handled by the service flow.

7.23.1.2 output_address_ext

`hsm_addr_msb_t open_svc_sign_ver_args_t::output_address_ext`

most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the commands handled by the service flow.

7.23.1.3 flags

`hsm_svc_signature_verification_flags_t open_svc_sign_ver_args_t::flags`

bitmap indicating the service flow properties

7.23.1.4 rsv

`uint8_t open_svc_sign_ver_args_t::rsv[3]`

Index

algo
 op_hash_one_go_args_t, 53
authentication_nonce
 open_svc_key_store_args_t, 63

ca_key
 hsm_op_pub_key_rec_args_t, 40
ca_key_ext
 hsm_op_pub_key_rec_args_t, 40
ca_key_size
 hsm_op_pub_key_rec_args_t, 41
cipher_algo
 op_cipher_one_go_args_t, 46

data1
 op_but_key_exp_args_t, 43
data1_size
 op_but_key_exp_args_t, 43
data2
 op_but_key_exp_args_t, 43
data2_size
 op_but_key_exp_args_t, 43
data3
 op_but_key_exp_args_t, 43
data3_size
 op_but_key_exp_args_t, 43
dest_key_identifier
 op_but_key_exp_args_t, 44

flags
 hsm_op_ecies_dec_args_t, 33
 hsm_op_ecies_enc_args_t, 36
 hsm_op_pub_key_dec_args_t, 38
 hsm_op_pub_key_rec_args_t, 41
 op_but_key_exp_args_t, 43
 op_cipher_one_go_args_t, 46
 op_finalize_sign_args_t, 48
 op_generate_key_args_t, 49
 op_generate_sign_args_t, 51
 op_hash_one_go_args_t, 53
 op_import_public_key_args_t, 54
 op_manage_key_args_t, 55
 op_prepare_sign_args_t, 57
 op_verify_sign_args_t, 59
 open_svc_cipher_args_t, 60
 open_svc_hash_args_t, 61
 open_svc_key_management_args_t, 62
 open_svc_key_store_args_t, 63
 open_svc_rng_args_t, 64
 open_svc_sign_gen_args_t, 65
 open_svc_sign_ver_args_t, 66

hash
 hsm_op_pub_key_rec_args_t, 40
hash_ext
 hsm_op_pub_key_rec_args_t, 40

hash_size
 hsm_op_pub_key_rec_args_t, 41
hsm_addr_lsb_t
 Hsm_api, 16
hsm_addr_msb_t
 Hsm_api, 16
Hsm_api, 3
 hsm_addr_lsb_t, 16
 hsm_addr_msb_t, 16
 hsm_butterfly_key_expansion, 21
 hsm_cipher_one_go, 22
 HSM_CIPHER_ONE_GO_ALGO_AES_CBC, 10
 HSM_CIPHER_ONE_GO_ALGO_AES_CCM, 10
 HSM_CIPHER_ONE_GO_ALGO_AES_ECB, 10
 HSM_CIPHER_ONE_GO_FLAGS_DECRYPT, 10
 HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT, 10
 hsm_close_cipher_service, 23
 hsm_close_hash_service, 29
 hsm_close_key_management_service, 21
 hsm_close_key_store_service, 19
 hsm_close_rng_service, 27
 hsm_close_session, 17
 hsm_close_signature_generation_service, 24
 hsm_close_signature_verification_service, 27
 HSM_CMD_NOT_SUPPORTED, 17
 hsm_ecies_decryption, 23
 hsm_ecies_encryption, 30
 hsm_err_t, 16
 hsm_finalize_signature, 25
 HSM_GENERAL_ERROR, 17
 hsm_generate_key, 20
 hsm_generate_signature, 24
 hsm_get_random, 28
 HSM_HASH_ALGO_SHA_224, 13
 HSM_HASH_ALGO_SHA_256, 13
 HSM_HASH_ALGO_SHA_384, 13
 hsm_hash_algo_t, 16
 hsm_hash_one_go, 29
 hsm_hdl_t, 13
 HSM_ID_CONFLICT, 17
 hsm_import_public_key, 26
 HSM_INVALID_ADDRESS, 17
 HSM_INVALID_LIFECYCLE, 17
 HSM_INVALID_MESSAGE, 17
 HSM_INVALID_PARAM, 17
 HSM_KEY_INFO_PERMANENT, 8
 hsm_key_info_t, 16
 HSM_KEY_STORAGE_ERROR, 17
 HSM_KEY_STORE_AUTH, 17
 HSM_KEY_STORE_CONFLICT, 17
 HSM_KEY_TYPE_AES_128, 8
 HSM_KEY_TYPE_AES_192, 8
 HSM_KEY_TYPE_AES_256, 8
 HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_224,
 7

HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_256, 7
 HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_384, 8
 HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_224, 8
 HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_256, 8
 HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_384, 8
 HSM_KEY_TYPE_ECDSA_NIST_P224, 7
 HSM_KEY_TYPE_ECDSA_NIST_P256, 7
 HSM_KEY_TYPE_ECDSA_NIST_P384, 7
 hsm_key_type_ext_t, 16
 hsm_key_type_t, 16
 hsm_manage_key, 20
 HSM_NO_ERROR, 17
 HSM_NVM_ERROR, 17
 hsm_op_but_key_exp_flags_t, 14
 hsm_op_cipher_one_go_algo_t, 14
 hsm_op_cipher_one_go_flags_t, 14
 hsm_op_ecies_dec_flags_t, 15
 hsm_op_ecies_enc_flags_t, 15
 HSM_OP_FINALIZE_SIGN_COMPRESSED_POINT, 12
 hsm_op_finalize_sign_flags_t, 15
 HSM_OP_FINALIZE_SIGN_INPUT_DIGEST, 12
 HSM_OP_FINALIZE_SIGN_INPUT_MESSAGE, 12
 HSM_OP_GENERATE_SIGN_COMPRESSED_POINT, 11
 hsm_op_generate_sign_flags_t, 15
 HSM_OP_GENERATE_SIGN_INPUT_DIGEST, 10
 HSM_OP_GENERATE_SIGN_INPUT_MESSAGE, 10
 hsm_op_hash_one_go_flags_t, 15
 hsm_op_key_gen_flags_t, 14
 HSM_OP_KEY_GENERATION_FLAGS_CREATE_PERSISTENT, 9
 HSM_OP_KEY_GENERATION_FLAGS_CREATE_TRANSIENT, 9
 HSM_OP_KEY_GENERATION_FLAGS_STRICT_OPERATION, 9
 HSM_OP_KEY_GENERATION_FLAGS_UPDATE, 8
 HSM_OP_MANAGE_KEY_FLAGS_CREATE_PERSISTENT, 9
 HSM_OP_MANAGE_KEY_FLAGS_CREATE_TRANSIENT, 9
 HSM_OP_MANAGE_KEY_FLAGS_DELETE, 9
 HSM_OP_MANAGE_KEY_FLAGS_STRICT_OPERATION, 9
 hsm_op_manage_key_flags_t, 14
 HSM_OP_MANAGE_KEY_FLAGS_UPDATE, 9
 hsm_op_prepare_signature_flags_t, 15
 hsm_op_pub_key_dec_flags_t, 15
 hsm_op_pub_key_rec_flags_t, 15
 HSM_OP_VERIFY_SIGN_COMPRESSED_POINT, 12
 hsm_op_verify_sign_flags_t, 15
 HSM_OP_VERIFY_SIGN_INPUT_DIGEST, 12
 HSM_OP_VERIFY_SIGN_INPUT_MESSAGE, 12
 HSM_OP_VERIFY_SIGN_KEY_INTERNAL, 12
 hsm_open_cipher_service, 22
 hsm_open_hash_service, 28
 hsm_open_key_management_service, 19
 hsm_open_key_store_service, 19
 hsm_open_rng_service, 27
 hsm_open_session, 17
 hsm_open_signature_generation_service, 23
 hsm_open_signature_verification_service, 25
 HSM_OUT_OF_MEMORY, 17
 hsm_prepare_signature, 24
 hsm_pub_key_decompression, 30
 hsm_pub_key_reconstruction, 29
 HSM_RNG_NOT_STARTED, 17
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_224_SHA, 11
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_256_SHA, 11
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_384_SHA, 11
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_224_SHA, 11
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_256_SHA, 11
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_384_SHA, 12
 HSM_SIGNATURE_SCHEME_ECDSA_NIST_P224_SHA_256, 11
 HSM_SIGNATURE_SCHEME_ECDSA_NIST_P256_SHA_256, 11
 HSM_SIGNATURE_SCHEME_ECDSA_NIST_P384_SHA_384, 11
 hsm_signature_scheme_id_t, 16
 hsm_svc_cipher_flags_t, 13
 hsm_svc_fast_signature_verification_flags_t, 14
 hsm_svc_hash_flags_t, 14
 hsm_svc_key_management_flags_t, 13
 HSM_SVC_KEY_STORE_FLAGS_CREATE, 7
 HSM_SVC_KEY_STORE_FLAGS_DELETE, 7
 hsm_svc_key_store_flags_t, 13
 HSM_SVC_KEY_STORE_FLAGS_UPDATE, 7
 hsm_svc_rng_flags_t, 14
 hsm_svc_signature_generation_flags_t, 13
 hsm_svc_signature_verification_flags_t, 14
 HSM_UNKNOWN_HANDLE, 17
 HSM_UNKNOWN_ID, 17
 HSM_UNKNOWN_KEY_STORE, 17
 HSM_VERIFICATION_STATUS_SUCCESS, 12
 hsm_verification_status_t, 16
 hsm_verify_signature, 26
 hsm_butterfly_key_expansion
 Hsm_api, 21
 hsm_cipher_one_go

Hsm_api, [22](#)
HSM_CIPHER_ONE_GO_ALGO_AES_CBC
Hsm_api, [10](#)
HSM_CIPHER_ONE_GO_ALGO_AES_CCM
Hsm_api, [10](#)
HSM_CIPHER_ONE_GO_ALGO_AES_ECB
Hsm_api, [10](#)
HSM_CIPHER_ONE_GO_FLAGS_DECRYPT
Hsm_api, [10](#)
HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT
Hsm_api, [10](#)
hsm_close_cipher_service
Hsm_api, [23](#)
hsm_close_hash_service
Hsm_api, [29](#)
hsm_close_key_management_service
Hsm_api, [21](#)
hsm_close_key_store_service
Hsm_api, [19](#)
hsm_close_rng_service
Hsm_api, [27](#)
hsm_close_session
Hsm_api, [17](#)
hsm_close_signature_generation_service
Hsm_api, [24](#)
hsm_close_signature_verification_service
Hsm_api, [27](#)
HSM_CMD_NOT_SUPPORTED
Hsm_api, [17](#)
hsm_ecies_decryption
Hsm_api, [23](#)
hsm_ecies_encryption
Hsm_api, [30](#)
hsm_err_t
Hsm_api, [16](#)
hsm_finalize_signature
Hsm_api, [25](#)
HSM_GENERAL_ERROR
Hsm_api, [17](#)
hsm_generate_key
Hsm_api, [20](#)
hsm_generate_signature
Hsm_api, [24](#)
hsm_get_random
Hsm_api, [28](#)
HSM_HASH_ALGO_SHA_224
Hsm_api, [13](#)
HSM_HASH_ALGO_SHA_256
Hsm_api, [13](#)
HSM_HASH_ALGO_SHA_384
Hsm_api, [13](#)
hsm_hash_algo_t
Hsm_api, [16](#)
hsm_hash_one_go
Hsm_api, [29](#)
hsm_hdl_t
Hsm_api, [13](#)
HSM_ID_CONFLICT
Hsm_api, [17](#)
hsm_import_public_key
Hsm_api, [26](#)
HSM_INVALID_ADDRESS
Hsm_api, [17](#)
HSM_INVALID_LIFECYCLE
Hsm_api, [17](#)
HSM_INVALID_MESSAGE
Hsm_api, [17](#)
HSM_INVALID_PARAM
Hsm_api, [17](#)
HSM_KEY_INFO_PERMANENT
Hsm_api, [8](#)
hsm_key_info_t
Hsm_api, [16](#)
HSM_KEY_STORAGE_ERROR
Hsm_api, [17](#)
HSM_KEY_STORE_AUTH
Hsm_api, [17](#)
HSM_KEY_STORE_CONFLICT
Hsm_api, [17](#)
HSM_KEY_TYPE_AES_128
Hsm_api, [8](#)
HSM_KEY_TYPE_AES_192
Hsm_api, [8](#)
HSM_KEY_TYPE_AES_256
Hsm_api, [8](#)
HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_224
Hsm_api, [7](#)
HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_256
Hsm_api, [7](#)
HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_384
Hsm_api, [8](#)
HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_224
Hsm_api, [8](#)
HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_256
Hsm_api, [8](#)
HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_384
Hsm_api, [8](#)
HSM_KEY_TYPE_ECDSA_NIST_P224
Hsm_api, [7](#)
HSM_KEY_TYPE_ECDSA_NIST_P256
Hsm_api, [7](#)
HSM_KEY_TYPE_ECDSA_NIST_P384
Hsm_api, [7](#)
hsm_key_type_ext_t
Hsm_api, [16](#)
hsm_key_type_t
Hsm_api, [16](#)
hsm_manage_key
Hsm_api, [20](#)
HSM_NO_ERROR
Hsm_api, [17](#)
HSM_NVM_ERROR
Hsm_api, [17](#)
hsm_op_but_key_exp_flags_t
Hsm_api, [14](#)
hsm_op_cipher_one_go_algo_t

- Hsm_api, 14
- hsm_op_cipher_one_go_flags_t
 - Hsm_api, 14
- hsm_op_ecies_dec_args_t, 30
 - flags, 33
 - input, 31
 - input_size, 32
 - key_identifier, 31
 - key_type, 32
 - mac_size, 32
 - output, 32
 - output_size, 32
 - p1, 31
 - p1_size, 32
 - p2, 31
 - p2_size, 32
- hsm_op_ecies_dec_flags_t
 - Hsm_api, 15
- hsm_op_ecies_enc_args_t, 33
 - flags, 36
 - input, 34
 - input_ext, 34
 - input_size, 35
 - key_type, 36
 - mac_size, 36
 - out_size, 36
 - output, 35
 - output_ext, 35
 - p1, 34
 - p1_ext, 34
 - p1_size, 35
 - p2, 35
 - p2_ext, 34
 - p2_size, 35
 - pub_key, 34
 - pub_key_ext, 34
 - pub_key_size, 35
 - rsv, 36
- hsm_op_ecies_enc_flags_t
 - Hsm_api, 15
- HSM_OP_FINALIZE_SIGN_COMPRESSED_POINT
 - Hsm_api, 12
- hsm_op_finalize_sign_flags_t
 - Hsm_api, 15
- HSM_OP_FINALIZE_SIGN_INPUT_DIGEST
 - Hsm_api, 12
- HSM_OP_FINALIZE_SIGN_INPUT_MESSAGE
 - Hsm_api, 12
- HSM_OP_GENERATE_SIGN_COMPRESSED_POINT
 - Hsm_api, 11
- hsm_op_generate_sign_flags_t
 - Hsm_api, 15
- HSM_OP_GENERATE_SIGN_INPUT_DIGEST
 - Hsm_api, 10
- HSM_OP_GENERATE_SIGN_INPUT_MESSAGE
 - Hsm_api, 10
- hsm_op_hash_one_go_flags_t
 - Hsm_api, 15
- hsm_op_key_gen_flags_t
 - Hsm_api, 14
- HSM_OP_KEY_GENERATION_FLAGS_CREATE_PERSISTENT
 - Hsm_api, 9
- HSM_OP_KEY_GENERATION_FLAGS_CREATE_TRANSIENT
 - Hsm_api, 9
- HSM_OP_KEY_GENERATION_FLAGS_STRICT_OPERATION
 - Hsm_api, 9
- HSM_OP_KEY_GENERATION_FLAGS_UPDATE
 - Hsm_api, 8
- HSM_OP_MANAGE_KEY_FLAGS_CREATE_PERSISTENT
 - Hsm_api, 9
- HSM_OP_MANAGE_KEY_FLAGS_CREATE_TRANSIENT
 - Hsm_api, 9
- HSM_OP_MANAGE_KEY_FLAGS_DELETE
 - Hsm_api, 9
- HSM_OP_MANAGE_KEY_FLAGS_STRICT_OPERATION
 - Hsm_api, 9
- hsm_op_manage_key_flags_t
 - Hsm_api, 14
- HSM_OP_MANAGE_KEY_FLAGS_UPDATE
 - Hsm_api, 9
- hsm_op_prepare_signature_flags_t
 - Hsm_api, 15
- hsm_op_pub_key_dec_args_t, 37
 - flags, 38
 - key, 37
 - key_ext, 37
 - key_size, 38
 - key_type, 38
 - out_key, 38
 - out_key_ext, 38
 - out_key_size, 38
 - rsv, 39
- hsm_op_pub_key_dec_flags_t
 - Hsm_api, 15
- hsm_op_pub_key_rec_args_t, 39
 - ca_key, 40
 - ca_key_ext, 40
 - ca_key_size, 41
 - flags, 41
 - hash, 40
 - hash_ext, 40
 - hash_size, 41
 - key_type, 41
 - out_key, 41
 - out_key_ext, 40
 - out_key_size, 41
 - pub_rec, 40
 - pub_rec_ext, 40
 - pub_rec_size, 41
 - rsv, 42
- hsm_op_pub_key_rec_flags_t
 - Hsm_api, 15
- HSM_OP_VERIFY_SIGN_COMPRESSED_POINT
 - Hsm_api, 12
- hsm_op_verify_sign_flags_t
 - Hsm_api, 15

HSM_OP_VERIFY_SIGN_INPUT_DIGEST
 Hsm_api, [12](#)
 HSM_OP_VERIFY_SIGN_INPUT_MESSAGE
 Hsm_api, [12](#)
 HSM_OP_VERIFY_SIGN_KEY_INTERNAL
 Hsm_api, [12](#)
 hsm_open_cipher_service
 Hsm_api, [22](#)
 hsm_open_hash_service
 Hsm_api, [28](#)
 hsm_open_key_management_service
 Hsm_api, [19](#)
 hsm_open_key_store_service
 Hsm_api, [19](#)
 hsm_open_rng_service
 Hsm_api, [27](#)
 hsm_open_session
 Hsm_api, [17](#)
 hsm_open_signature_generation_service
 Hsm_api, [23](#)
 hsm_open_signature_verification_service
 Hsm_api, [25](#)
 HSM_OUT_OF_MEMORY
 Hsm_api, [17](#)
 hsm_prepare_signature
 Hsm_api, [24](#)
 hsm_pub_key_decompression
 Hsm_api, [30](#)
 hsm_pub_key_reconstruction
 Hsm_api, [29](#)
 HSM_RNG_NOT_STARTED
 Hsm_api, [17](#)
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_224_SHA_256
 Hsm_api, [11](#)
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_256_SHA_256
 Hsm_api, [11](#)
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_R1_384_SHA_384
 Hsm_api, [11](#)
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_224_SHA_256
 Hsm_api, [11](#)
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_256_SHA_256
 Hsm_api, [11](#)
 HSM_SIGNATURE_SCHEME_ECDSA_BRAINPOOL_T1_384_SHA_384
 Hsm_api, [12](#)
 HSM_SIGNATURE_SCHEME_ECDSA_NIST_P224_SHA_256
 Hsm_api, [11](#)
 HSM_SIGNATURE_SCHEME_ECDSA_NIST_P256_SHA_256
 Hsm_api, [11](#)
 HSM_SIGNATURE_SCHEME_ECDSA_NIST_P384_SHA_384
 Hsm_api, [11](#)
 hsm_signature_scheme_id_t
 Hsm_api, [16](#)
 hsm_svc_cipher_flags_t
 Hsm_api, [13](#)
 hsm_svc_fast_signature_verification_flags_t
 Hsm_api, [14](#)
 hsm_svc_hash_flags_t
 Hsm_api, [14](#)
 hsm_svc_key_management_flags_t
 Hsm_api, [13](#)
 HSM_SVC_KEY_STORE_FLAGS_CREATE
 Hsm_api, [7](#)
 HSM_SVC_KEY_STORE_FLAGS_DELETE
 Hsm_api, [7](#)
 hsm_svc_key_store_flags_t
 Hsm_api, [13](#)
 HSM_SVC_KEY_STORE_FLAGS_UPDATE
 Hsm_api, [7](#)
 hsm_svc_rng_flags_t
 Hsm_api, [14](#)
 hsm_svc_signature_generation_flags_t
 Hsm_api, [13](#)
 hsm_svc_signature_verification_flags_t
 Hsm_api, [14](#)
 HSM_UNKNOWN_HANDLE
 Hsm_api, [17](#)
 HSM_UNKNOWN_ID
 Hsm_api, [17](#)
 HSM_UNKNOWN_KEY_STORE
 Hsm_api, [17](#)
 HSM_VERIFICATION_STATUS_SUCCESS
 Hsm_api, [12](#)
 hsm_verification_status_t
 Hsm_api, [16](#)
 hsm_verify_signature
 Hsm_api, [26](#)
 input
 hsm_op_ecies_dec_args_t, [31](#)
 hsm_op_ecies_enc_args_t, [34](#)
 op_cipher_one_go_args_t, [46](#)
 op_hash_one_go_args_t, [52](#)
 input_address_ext
 open_svc_cipher_args_t, [60](#)
 open_svc_hash_args_t, [61](#)
 open_svc_key_management_args_t, [62](#)
 open_svc_rng_args_t, [64](#)
 open_svc_sign_gen_args_t, [65](#)
 open_svc_sign_ver_args_t, [66](#)
 input_ext
 hsm_op_ecies_enc_args_t, [34](#)
 input_key
 op_manage_key_args_t, [56](#)
 input_size
 hsm_op_ecies_dec_args_t, [32](#)
 hsm_op_ecies_enc_args_t, [35](#)
 op_cipher_one_go_args_t, [46](#)
 op_hash_one_go_args_t, [53](#)
 op_manage_key_args_t, [55](#)
 iv
 op_cipher_one_go_args_t, [45](#)
 iv_size
 op_cipher_one_go_args_t, [45](#)
 key
 hsm_op_pub_key_dec_args_t, [37](#)
 op_import_public_key_args_t, [54](#)

- op_verify_sign_args_t, 57
- key_ext
 - hsm_op_pub_key_dec_args_t, 37
- key_identifier
 - hsm_op_ecies_dec_args_t, 31
 - op_but_key_exp_args_t, 42
 - op_cipher_one_go_args_t, 45
 - op_finalize_sign_args_t, 47
 - op_generate_key_args_t, 49
 - op_generate_sign_args_t, 50
 - op_manage_key_args_t, 55
- key_info
 - op_generate_key_args_t, 49
 - op_manage_key_args_t, 56
- key_size
 - hsm_op_pub_key_dec_args_t, 38
 - op_import_public_key_args_t, 54
 - op_verify_sign_args_t, 58
- key_store_identifier
 - open_svc_key_store_args_t, 63
- key_type
 - hsm_op_ecies_dec_args_t, 32
 - hsm_op_ecies_enc_args_t, 36
 - hsm_op_pub_key_dec_args_t, 38
 - hsm_op_pub_key_rec_args_t, 41
 - op_but_key_exp_args_t, 44
 - op_generate_key_args_t, 49
 - op_import_public_key_args_t, 54
 - op_manage_key_args_t, 56
- key_type_ext
 - op_generate_key_args_t, 49
 - op_manage_key_args_t, 56
- mac_size
 - hsm_op_ecies_dec_args_t, 32
 - hsm_op_ecies_enc_args_t, 36
- max_updates_number
 - open_svc_key_store_args_t, 63
- message
 - op_finalize_sign_args_t, 47
 - op_generate_sign_args_t, 50
 - op_verify_sign_args_t, 58
- message_size
 - op_finalize_sign_args_t, 48
 - op_generate_sign_args_t, 51
 - op_verify_sign_args_t, 58
- op_but_key_exp_args_t, 42
 - data1, 43
 - data1_size, 43
 - data2, 43
 - data2_size, 43
 - data3, 43
 - data3_size, 43
 - dest_key_identifier, 44
 - flags, 43
 - key_identifier, 42
 - key_type, 44
 - output, 44
 - output_size, 44
 - rsv, 44
- op_cipher_one_go_args_t, 45
 - cipher_algo, 46
 - flags, 46
 - input, 46
 - input_size, 46
 - iv, 45
 - iv_size, 45
 - key_identifier, 45
 - output, 46
 - output_size, 46
- op_finalize_sign_args_t, 47
 - flags, 48
 - key_identifier, 47
 - message, 47
 - message_size, 48
 - rsv, 48
 - signature, 47
 - signature_size, 48
- op_generate_key_args_t, 48
 - flags, 49
 - key_identifier, 49
 - key_info, 49
 - key_type, 49
 - key_type_ext, 49
 - out_key, 50
 - out_size, 49
 - rsv, 49
- op_generate_sign_args_t, 50
 - flags, 51
 - key_identifier, 50
 - message, 50
 - message_size, 51
 - scheme_id, 51
 - signature, 51
 - signature_size, 51
- op_get_random_args_t, 52
 - output, 52
 - random_size, 52
- op_hash_one_go_args_t, 52
 - algo, 53
 - flags, 53
 - input, 52
 - input_size, 53
 - output, 53
 - output_size, 53
 - rsv, 53
- op_import_public_key_args_t, 54
 - flags, 54
 - key, 54
 - key_size, 54
 - key_type, 54
- op_manage_key_args_t, 55
 - flags, 55
 - input_key, 56
 - input_size, 55
 - key_identifier, 55

- key_info, 56
- key_type, 56
- key_type_ext, 56
- rsv, 55
- op_prepare_sign_args_t, 56
 - flags, 57
 - rsv, 57
 - scheme_id, 56
- op_verify_sign_args_t, 57
 - flags, 59
 - key, 57
 - key_size, 58
 - message, 58
 - message_size, 58
 - rsv, 59
 - scheme_id, 58
 - signature, 58
 - signature_size, 58
- open_session_args_t, 59
 - operating_mode, 59
 - rsv, 59
 - session_priority, 59
- open_svc_cipher_args_t, 60
 - flags, 60
 - input_address_ext, 60
 - output_address_ext, 60
 - rsv, 60
- open_svc_hash_args_t, 61
 - flags, 61
 - input_address_ext, 61
 - output_address_ext, 61
 - rsv, 61
- open_svc_key_management_args_t, 62
 - flags, 62
 - input_address_ext, 62
 - output_address_ext, 62
 - rsv, 62
- open_svc_key_store_args_t, 63
 - authentication_nonce, 63
 - flags, 63
 - key_store_identifier, 63
 - max_updates_number, 63
 - rsv, 63
- open_svc_rng_args_t, 64
 - flags, 64
 - input_address_ext, 64
 - output_address_ext, 64
 - rsv, 64
- open_svc_sign_gen_args_t, 65
 - flags, 65
 - input_address_ext, 65
 - output_address_ext, 65
 - rsv, 65
- open_svc_sign_ver_args_t, 66
 - flags, 66
 - input_address_ext, 66
 - output_address_ext, 66
 - rsv, 66
- operating_mode
 - open_session_args_t, 59
- out_key
 - hsm_op_pub_key_dec_args_t, 38
 - hsm_op_pub_key_rec_args_t, 41
 - op_generate_key_args_t, 50
- out_key_ext
 - hsm_op_pub_key_dec_args_t, 38
 - hsm_op_pub_key_rec_args_t, 40
- out_key_size
 - hsm_op_pub_key_dec_args_t, 38
 - hsm_op_pub_key_rec_args_t, 41
- out_size
 - hsm_op_ecies_enc_args_t, 36
 - op_generate_key_args_t, 49
- output
 - hsm_op_ecies_dec_args_t, 32
 - hsm_op_ecies_enc_args_t, 35
 - op_but_key_exp_args_t, 44
 - op_cipher_one_go_args_t, 46
 - op_get_random_args_t, 52
 - op_hash_one_go_args_t, 53
- output_address_ext
 - open_svc_cipher_args_t, 60
 - open_svc_hash_args_t, 61
 - open_svc_key_management_args_t, 62
 - open_svc_rng_args_t, 64
 - open_svc_sign_gen_args_t, 65
 - open_svc_sign_ver_args_t, 66
- output_ext
 - hsm_op_ecies_enc_args_t, 35
- output_size
 - hsm_op_ecies_dec_args_t, 32
 - op_but_key_exp_args_t, 44
 - op_cipher_one_go_args_t, 46
 - op_hash_one_go_args_t, 53
- p1
 - hsm_op_ecies_dec_args_t, 31
 - hsm_op_ecies_enc_args_t, 34
- p1_ext
 - hsm_op_ecies_enc_args_t, 34
- p1_size
 - hsm_op_ecies_dec_args_t, 32
 - hsm_op_ecies_enc_args_t, 35
- p2
 - hsm_op_ecies_dec_args_t, 31
 - hsm_op_ecies_enc_args_t, 35
- p2_ext
 - hsm_op_ecies_enc_args_t, 34
- p2_size
 - hsm_op_ecies_dec_args_t, 32
 - hsm_op_ecies_enc_args_t, 35
- pub_key
 - hsm_op_ecies_enc_args_t, 34
- pub_key_ext
 - hsm_op_ecies_enc_args_t, 34
- pub_key_size
 - hsm_op_ecies_enc_args_t, 35

- pub_rec
 - hsm_op_pub_key_rec_args_t, [40](#)
- pub_rec_ext
 - hsm_op_pub_key_rec_args_t, [40](#)
- pub_rec_size
 - hsm_op_pub_key_rec_args_t, [41](#)
- random_size
 - op_get_random_args_t, [52](#)
- rsv
 - hsm_op_ecies_enc_args_t, [36](#)
 - hsm_op_pub_key_dec_args_t, [39](#)
 - hsm_op_pub_key_rec_args_t, [42](#)
 - op_but_key_exp_args_t, [44](#)
 - op_finalize_sign_args_t, [48](#)
 - op_generate_key_args_t, [49](#)
 - op_hash_one_go_args_t, [53](#)
 - op_manage_key_args_t, [55](#)
 - op_prepare_sign_args_t, [57](#)
 - op_verify_sign_args_t, [59](#)
 - open_session_args_t, [59](#)
 - open_svc_cipher_args_t, [60](#)
 - open_svc_hash_args_t, [61](#)
 - open_svc_key_management_args_t, [62](#)
 - open_svc_key_store_args_t, [63](#)
 - open_svc_rng_args_t, [64](#)
 - open_svc_sign_gen_args_t, [65](#)
 - open_svc_sign_ver_args_t, [66](#)
- scheme_id
 - op_generate_sign_args_t, [51](#)
 - op_prepare_sign_args_t, [56](#)
 - op_verify_sign_args_t, [58](#)
- session_priority
 - open_session_args_t, [59](#)
- signature
 - op_finalize_sign_args_t, [47](#)
 - op_generate_sign_args_t, [51](#)
 - op_verify_sign_args_t, [58](#)
- signature_size
 - op_finalize_sign_args_t, [48](#)
 - op_generate_sign_args_t, [51](#)
 - op_verify_sign_args_t, [58](#)