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 1

3.1 Session . . . 1

3.2 Service flow . . . 1

4 Module Index 2

4.1 Modules . . . 2

5 Module Documentation 2

5.1 Hsm_api . . . 2

5.1.1 Detailed Description . . . 3

5.1.2 Macro Definition Documentation . . . 3

5.1.3 Enumeration Type Documentation . . . 5

5.1.4 Function Documentation . . . 6

Index 11

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

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 requester 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 requester.

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 2

5 Module Documentation

5.1 Hsm_api

i.MX8 HSM API header file

Macros

- #define HSM_KEY_STORAGE_ACCESS_FLAG_NEW (1 << 0)
- #define HSM_KEY_IDENTIFIER_NEW 0xFFFFFFFF
- #define HSM_KEY_TYPE_ECDSA_NIST_P224 0x00
- #define HSM_KEY_TYPE_ECDSA_NIST_P256 0x01
- #define HSM_KEY_TYPE_ECDSA_NIST_P384 0x02
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_224 0x10
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_256 0x11
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_384 0x12
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_224 0x20
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_256 0x21
- #define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_384 0x22
- #define HSM_KEY_TYPE_AES_128 0x30
- #define HSM_KEY_TYPE_AES_192 0x31
- #define HSM_KEY_TYPE_AES_256 0x32
- #define HSM_KEY_FLAGS_TRANSIENT (1 << 0)
- #define HSM_KEY_FLAGS_PERMANENT (1 << 1)
- #define HSM_CIPHER_ONE_GO_ALGO_AES_ECB 0x00
- #define HSM_CIPHER_ONE_GO_ALGO_AES_CBC 0x01
- #define HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT (1)
- #define HSM_CIPHER_ONE_GO_FLAGS_DECRYPT (0)

Enumerations

- enum hsm_err_t {
  HSM_NO_ERROR = 0x0,
  HSM_OUT_OF_MEM = 0x1,
  HSM_UNKNOWN_HANDLE = 0x2,
  HSM_UNKNOWN_KEY_STORE = 0x3,
  HSM_KEY_STORE_AUTH_ERROR = 0x4,
  HSM_UNKNOWN_ID = 0x5,
  HSM_BUF_SIZE_ERROR = 0x6,
  HSM_KEY_ERROR = 0x7,
  HSM_MEM_ACCESS_ERROR = 0x8,
  HSM_INVALID_PARAM = 0x9,
  HSM_GENERAL_ERROR = 0xFF }

  Error codes returned by HSM functions.

Functions

- struct hsm_hdl_s * hsm_open_session (uint32_t key_storage_identifier, uint8_t access_flags, uint32_t password, uint8_t session_priority, uint8_t operating_mode)
- hsm_err_t hsm_close_session (struct hsm_hdl_s *hdl)
- hsm_err_t hsm_open_key_management_service (struct hsm_hdl_s *hdl, uint32_t input_address_ext, uint32_t output_address_ext)
- hsm_err_t hsm_key_management_cmd_key_generation (struct hsm_hdl_s *hdl, uint8_t *key_identifier, uint8_t *output, uint16_t key_type, uint8_t output_size, uint8_t flags)
- hsm_err_t hsm_close_key_management_service (struct hsm_hdl_s *hdl)
- hsm_err_t hsm_open_cipher_service (struct hsm_hdl_s *hdl, uint32_t input_address_ext, uint32_t output_address_ext, uint8_t flags)
- hsm_err_t hsm_cipher_cmd_cipher_one_go (struct hsm_hdl_s *hdl, uint32_t key_identifier, uint8_t *input, uint8_t *output, uint8_t *iv, uint32_t *input_size, uint16_t iv_size, uint8_t algorithm, uint8_t flags)
- hsm_err_t hsm_close_cipher_service (struct hsm_hdl_s *hdl)

5.1.1 Detailed Description

i.MX8 HSM API header file

5.1.2 Macro Definition Documentation

5.1.2.1 HSM_KEY_STORAGE_ACCESS_FLAG_NEW

```
#define HSM_KEY_STORAGE_ACCESS_FLAG_NEW (1 << 0)
```

It must be specified to create a new key storage

5.1.2.2 HSM_KEY_IDENTIFIER_NEW

```
#define HSM_KEY_IDENTIFIER_NEW 0xFFFFFFFF
```

It must be specified to create a new key slot

5.1.2.3 HSM_KEY_TYPE_ECDSA_NIST_P224

```
#define HSM_KEY_TYPE_ECDSA_NIST_P224 0x00
```

5.1.2.4 HSM_KEY_TYPE_ECDSA_NIST_P256

```
#define HSM_KEY_TYPE_ECDSA_NIST_P256 0x01
```

5.1.2.5 HSM_KEY_TYPE_ECDSA_NIST_P384

```
#define HSM_KEY_TYPE_ECDSA_NIST_P384 0x02
```

5.1.2.6 HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_224

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_224 0x10
```

5.1.2.7 HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_256

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_256 0x11
```

5.1.2.8 HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_384

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_384 0x12
```

5.1.2.9 HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_224

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_224 0x20
```

5.1.2.10 HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_256

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_256 0x21
```

5.1.2.11 HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_384

```
#define HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_384 0x22
```

5.1.2.12 HSM_KEY_TYPE_AES_128

```
#define HSM_KEY_TYPE_AES_128 0x30
```

5.1.2.13 HSM_KEY_TYPE_AES_192

```
#define HSM_KEY_TYPE_AES_192 0x31
```

5.1.2.14 HSM_KEY_TYPE_AES_256

```
#define HSM_KEY_TYPE_AES_256 0x32
```

5.1.2.15 HSM_KEY_FLAGS_TRANSIENT

```
#define HSM_KEY_FLAGS_TRANSIENT (1 << 0)
```

When set, the key is transient. Transient keys are deleted when the corresponding key store service flow is closed.

5.1.2.16 HSM_KEY_FLAGS_PERMANENT

```
#define HSM_KEY_FLAGS_PERMANENT (1 << 1)
```

When set, the key is permanent. Once created, it will not be possible to update or delete the key anymore.

5.1.2.17 HSM_CIPHER_ONE_GO_ALGO_AES_ECB

```
#define HSM_CIPHER_ONE_GO_ALGO_AES_ECB 0x00
```

5.1.2.18 HSM_CIPHER_ONE_GO_ALGO_AES_CBC

```
#define HSM_CIPHER_ONE_GO_ALGO_AES_CBC 0x01
```

5.1.2.19 HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT

```
#define HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT (1)
```

5.1.2.20 HSM_CIPHER_ONE_GO_FLAGS_DECRYPT

```
#define HSM_CIPHER_ONE_GO_FLAGS_DECRYPT (0)
```

5.1.3 Enumeration Type Documentation

5.1.3.1 hsm_err_t

```
enum hsm_err_t
```

Error codes returned by HSM functions.

Enumerator

HSM_NO_ERROR	Success.
HSM_OUT_OF_MEM	There is not enough memory to open a new session or service flow.
HSM_UNKNOWN_HANDLE	The provided handle doens't exist
HSM_UNKNOWN_KEY_STORE	The provided key store identifier doesn't exist
HSM_KEY_STORE_AUTH_ERROR	Key store authentication fails
HSM_UNKNOWN_ID	The provided identifier doens't exist.
HSM_BUF_SIZE_ERROR	The size of the buffer provided by the requester is too small for the requested operation
HSM_KEY_ERROR	The key cannot be used for the requested opearation.
HSM_MEM_ACCESS_ERROR	The specified memory address cannot be accessed.
HSM_INVALID_PARAM	One or more parameters are not valid
HSM_GENERAL_ERROR	Error not covered by other codes occured.

5.1.4 Function Documentation

5.1.4.1 hsm_open_session()

```
struct hsm_hdl_s* hsm_open_session (
            uint32_t key_storage_identifier,
            uint8_t access_flags,
            uint32_t password,
            uint8_t session_priority,
            uint8_t operating_mode )
```

Initiate a HSM session granting the usage of the specified key store.
The returned handle pointer is typed with the transparent struct "hsm_hdl_s". The user doesn't need to know or to access the fields of this struct. They only need to store this pointer and pass it to every calls to other APIs within the same HSM session.

Parameters

key_storage_identifier	key store identifier
access_flags	bitmap indicating the requested access to the key store. The create flag must be specified to create a new key storage.
password	password for accesing the key storage
session_priority	not supported in current release, any value accepted.
operating_mode	not supported in current release, any value accepted.

Returns

pointer to the HSM handle.

5.1.4.2 hsm_close_session()

```
hsm_err_t hsm_close_session (
            struct hsm_hdl_s * hdl )
```

Terminate a previously opened HSM session

Parameters

hdl	pointer to the HSM handle to be closed.

Returns

error code

5.1.4.3 hsm_open_key_management_service()

```
hsm_err_t hsm_open_key_management_service (
            struct hsm_hdl_s * hdl,
            uint32_t input_address_ext,
            uint32_t output_address_ext )
```

Open a key management service flow
User must open this service in order to perform operation on the keys (generate, delete, update)

Parameters

hdl	pointer to the HSM handle
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.
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.

Returns

error code

5.1.4.4 hsm_key_management_cmd_key_generation()

```
hsm_err_t hsm_key_management_cmd_key_generation (
            struct hsm_hdl_s * hdl,
            uint8_t * key_identifier,
            uint8_t * output,
            uint16_t key_type,
            uint8_t output_size,
            uint8_t flags )
```

Generate a key or a key pair in the key store. The public key can optionally be exported
User can call this function only after having opened a key management service flow

Parameters

hdl	pointer to the HSM handle
key_identifier	pointer to the identifier of the key slot to be used for the operation - The value HSM_KEY_IDENTIFIER_NEW indicates to create a new key slot
output	pointer to the output area to store the public key - A NULL pointer indicates to not store the public key
key_type	indicates which type of key must be generated
output_size	lenght in bytes of the output area
flags	bitmap specifying the properties of the key

Returns

error code

5.1.4.5 hsm_close_key_management_service()

```
hsm_err_t hsm_close_key_management_service (
            struct hsm_hdl_s * hdl )
```

Terminate a previously opened key management service flow

Parameters

hdl	pointer to the HSM handle.

Returns

error code

5.1.4.6 hsm_open_cipher_service()

```
hsm_err_t hsm_open_cipher_service (
            struct hsm_hdl_s * hdl,
            uint32_t input_address_ext,
            uint32_t output_address_ext,
            uint8_t flags )
```

Open a cipher service flow
User must open this service in order to perform cipher operations.

Parameters

hdl	pointer to the HSM handle
input_address_ext	most significant 32 bits address to be used by HSM for input memory transactions in the requester address space for the operations handled by the service flow.
output_address_ext	most significant 32 bits address to be used by HSM for output memory transactions in the requester address space for the opeartion handled by the service flow.
flags	bitmap indicating the service flow properties - not supported in current release, any value accepted.

Returns

error code

5.1.4.7 hsm_cipher_cmd_cipher_one_go()

```
hsm_err_t hsm_cipher_cmd_cipher_one_go (
            struct hsm_hdl_s * hdl,
            uint32_t key_identifier,
            uint8_t * input,
            uint8_t * output,
            uint8_t * iv,
            uint32_t * input_size,
            uint16_t iv_size,
            uint8_t algorithm,
            uint8_t flags )
```

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

Parameters

hdl	pointer to the HSM handle
key_identifier	identifier of the key to be used for the operation
input	pointer to the input to be processed
output	pointer to the output area
iv	pointer to the initialization vector - it must be NULL for algorithms not using the initialization vector
input_size	lenght in bytes of the input
iv_size	lenght in bytes of the initialization vector - it must be 0 for algorithms not using the initialization vector
algorithm	to be used for the operation
flags	bitmap specifying the operation attributes

Returns

error code

5.1.4.8 hsm_close_cipher_service()

```
hsm_err_t hsm_close_cipher_service (
            struct hsm_hdl_s * hdl )
```

Terminate a previously opened cipher service flow

Parameters

hdl	pointer to the HSM handle.

Returns

error code

Index

Hsm_api, 2
HSM_BUF_SIZE_ERROR, 6
hsm_cipher_cmd_cipher_one_go, 9
HSM_CIPHER_ONE_GO_ALGO_AES_CBC, 5
HSM_CIPHER_ONE_GO_ALGO_AES_ECB, 5
HSM_CIPHER_ONE_GO_FLAGS_DECRYPT, 5
HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT, 5
hsm_close_cipher_service, 9
hsm_close_key_management_service, 8
hsm_close_session, 6
hsm_err_t, 5
HSM_GENERAL_ERROR, 6
HSM_INVALID_PARAM, 6
HSM_KEY_ERROR, 6
HSM_KEY_FLAGS_PERMANENT, 5
HSM_KEY_FLAGS_TRANSIENT, 5
HSM_KEY_IDENTIFIER_NEW, 3
hsm_key_management_cmd_key_generation, 7
HSM_KEY_STORAGE_ACCESS_FLAG_NEW, 3
HSM_KEY_STORE_AUTH_ERROR, 6
HSM_KEY_TYPE_AES_128, 4
HSM_KEY_TYPE_AES_192, 4
HSM_KEY_TYPE_AES_256, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_224, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_256, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_384, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_224, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_256, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_384, 4
HSM_KEY_TYPE_ECDSA_NIST_P224, 3
HSM_KEY_TYPE_ECDSA_NIST_P256, 3
HSM_KEY_TYPE_ECDSA_NIST_P384, 3
HSM_MEM_ACCESS_ERROR, 6
HSM_NO_ERROR, 6
hsm_open_cipher_service, 8
hsm_open_key_management_service, 7
hsm_open_session, 6
HSM_OUT_OF_MEM, 6
HSM_UNKNOWN_HANDLE, 6
HSM_UNKNOWN_ID, 6
HSM_UNKNOWN_KEY_STORE, 6
HSM_BUF_SIZE_ERROR
Hsm_api, 6
hsm_cipher_cmd_cipher_one_go
Hsm_api, 9
HSM_CIPHER_ONE_GO_ALGO_AES_CBC
Hsm_api, 5
HSM_CIPHER_ONE_GO_ALGO_AES_ECB
Hsm_api, 5
HSM_CIPHER_ONE_GO_FLAGS_DECRYPT
Hsm_api, 5
HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT
Hsm_api, 5
hsm_close_cipher_service
Hsm_api, 9
hsm_close_key_management_service
Hsm_api, 8
hsm_close_session
Hsm_api, 6
hsm_err_t
Hsm_api, 5
HSM_GENERAL_ERROR
Hsm_api, 6
HSM_INVALID_PARAM
Hsm_api, 6
HSM_KEY_ERROR
Hsm_api, 6
HSM_KEY_FLAGS_PERMANENT
Hsm_api, 5
HSM_KEY_FLAGS_TRANSIENT
Hsm_api, 5
HSM_KEY_IDENTIFIER_NEW
Hsm_api, 3
hsm_key_management_cmd_key_generation
Hsm_api, 7
HSM_KEY_STORAGE_ACCESS_FLAG_NEW
Hsm_api, 3
HSM_KEY_STORE_AUTH_ERROR
Hsm_api, 6
HSM_KEY_TYPE_AES_128
Hsm_api, 4
HSM_KEY_TYPE_AES_192
Hsm_api, 4
HSM_KEY_TYPE_AES_256
Hsm_api, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_224
Hsm_api, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_256
Hsm_api, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_R1_384
Hsm_api, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_224
Hsm_api, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_256
Hsm_api, 4
HSM_KEY_TYPE_ECDSA_BRAINPOOL_T1_384
Hsm_api, 4
HSM_KEY_TYPE_ECDSA_NIST_P224
Hsm_api, 3
HSM_KEY_TYPE_ECDSA_NIST_P256
Hsm_api, 3
HSM_KEY_TYPE_ECDSA_NIST_P384
Hsm_api, 3
HSM_MEM_ACCESS_ERROR

Hsm_api, 6
HSM_NO_ERROR
Hsm_api, 6
hsm_open_cipher_service
Hsm_api, 8
hsm_open_key_management_service
Hsm_api, 7
hsm_open_session
Hsm_api, 6
HSM_OUT_OF_MEM
Hsm_api, 6
HSM_UNKNOWN_HANDLE
Hsm_api, 6
HSM_UNKNOWN_ID
Hsm_api, 6
HSM_UNKNOWN_KEY_STORE
Hsm_api, 6