# i.MX8 SHE API

Revision_0.1

Generated by Doxygen 1.8.11

# Contents

# 1 Main Page

# 2 Module Index

## 2.1 Modules

Here is a list of all modules:

# 3 Module Documentation

## 3.1 She_api

SHE feature API.

**Macros**

- #define SHE_KEY_1 (0x04)

    *Identifiers for SHE keys.*

- #define SHE_KEY_2 (0x05)
- #define SHE_KEY_3 (0x06)
- #define SHE_KEY_4 (0x07)
- #define SHE_KEY_5 (0x08)
- #define SHE_KEY_6 (0x09)
- #define SHE_KEY_7 (0x0a)
- #define SHE_KEY_8 (0x0b)
- #define SHE_KEY_9 (0x0c)
- #define SHE_KEY_10 (0x0d)
- #define SHE_RAM_KEY (0x0e)
- #define SHE_KEY_DEFAULT (0x00)

    *Identifiers for SHE keys extensions.*

- #define SHE_KEY_N_EXT_1 (0x10)
- #define SHE_KEY_N_EXT_2 (0x20)
- #define SHE_KEY_N_EXT_3 (0x30)
- #define SHE_KEY_N_EXT_4 (0x40)
- #define SHE_STORAGE_CREATE_SUCCESS 0u
- #define SHE_STORAGE_CREATE_WARNING 1u
- #define SHE_STORAGE_CREATE_UNAUTHORIZED 2u
- #define SHE_STORAGE_CREATE_FAIL 3u
- #define SHE_STORAGE_NUMBER_UPDATES_DEFAULT 300u
- #define SHE_MAC_SIZE 16u
- #define SHE_MAC_VERIFICATION_SUCCESS 0u
- #define SHE_MAC_VERIFICATION_FAILED 1u
- #define SHE_AES_BLOCK_SIZE_128 16u
- #define SHE_KEY_SIZE 16u /** SHE keys are 128 bits (16 bytes) long. */
- #define SHE_ENTROPY_SIZE 16u
- #define SHE_RND_SIZE 16u
- #define SHE_CHALLENGE_SIZE 16u /* 128 bits */
- #define SHE_ID_SIZE 15u /* 120 bits */

**Enumerations**

- enum she_err_t {
  ERC_NO_ERROR = 0x0, ERC_SEQUENCE_ERROR = 0x1, ERC_KEY_NOT_AVAILABLE = 0x2, ERC_↩
  KEY_INVALID = 0x3,
  ERC_KEY_EMPTY = 0x4, ERC_NO_SECURE_BOOT = 0x5, ERC_KEY_WRITE_PROTECTED = 0x6, E↩
  RC_KEY_UPDATE_ERROR = 0x7,
  ERC_RNG_SEED = 0x8, ERC_NO_DEBUGGING = 0x9, ERC_BUSY = 0xA, ERC_MEMORY_FAILURE =
  0xB,
  ERC_GENERAL_ERROR = 0xC }

    *Error codes returned by SHE functions.*

**Functions**

- uint32_t she_storage_create (uint32_t key_storage_identifier, uint32_t password, uint16_t max_updates_↩
number, uint8_t ∗signed_message, uint32_t msg_len)
- struct she_hdl_s ∗ she_open_session (uint32_t key_storage_identifier, uint32_t password, void(∗async_↩
cb)(void ∗priv, she_err_t err), void ∗priv)
- void she_close_session (struct she_hdl_s ∗hdl)
- she_err_t she_cmd_generate_mac (struct she_hdl_s ∗hdl, uint8_t key_ext, uint8_t key_id, uint16_↩
t message_length, uint8_t ∗message, uint8_t ∗mac)
- she_err_t she_cmd_verify_mac (struct she_hdl_s ∗hdl, uint8_t key_ext, uint8_t key_id, uint16_t message_↩
length, uint8_t ∗message, uint8_t ∗mac, uint8_t mac_length, uint8_t ∗verification_status)
- she_err_t she_cmd_enc_cbc (struct she_hdl_s ∗hdl, uint8_t key_ext, uint8_t key_id, uint32_t data_length,
uint8_t ∗iv, uint8_t ∗plaintext, uint8_t ∗ciphertext)
- she_err_t she_cmd_dec_cbc (struct she_hdl_s ∗hdl, uint8_t key_ext, uint8_t key_id, uint32_t data_length,
uint8_t ∗iv, uint8_t ∗ciphertext, uint8_t ∗plaintext)
- she_err_t she_cmd_enc_ecb (struct she_hdl_s ∗hdl, uint8_t key_ext, uint8_t key_id, uint8_t ∗plaintext,
uint8_t ∗ciphertext)
- she_err_t she_cmd_dec_ecb (struct she_hdl_s ∗hdl, uint8_t key_ext, uint8_t key_id, uint8_t ∗ciphertext,
uint8_t ∗plaintext)
- she_err_t she_cmd_load_key (struct she_hdl_s ∗hdl, uint8_t key_ext, uint8_t key_id, uint8_t ∗m1, uint8_t
∗m2, uint8_t ∗m3, uint8_t ∗m4, uint8_t ∗m5)
- she_err_t she_cmd_load_plain_key (struct she_hdl_s ∗hdl, uint8_t ∗key)
- she_err_t she_cmd_export_ram_key (struct she_hdl_s ∗hdl, uint8_t ∗m1, uint8_t ∗m2, uint8_t ∗m3, uint8_t
∗m4, uint8_t ∗m5)
- she_err_t she_cmd_init_rng (struct she_hdl_s ∗hdl)
- she_err_t she_cmd_extend_seed (struct she_hdl_s ∗hdl, uint8_t ∗entropy)
- she_err_t she_cmd_rnd (struct she_hdl_s ∗hdl, uint8_t ∗rnd)
- she_err_t she_cmd_get_status (struct she_hdl_s ∗hdl, uint8_t ∗sreg)
- she_err_t she_cmd_get_id (struct she_hdl_s ∗hdl, uint8_t ∗challenge, uint8_t ∗id, uint8_t ∗sreg, uint8_↩
t ∗mac)
- she_err_t she_cmd_cancel (struct she_hdl_s ∗hdl)

### 3.1.1 Detailed Description

SHE feature API.

### 3.1.2 Macro Definition Documentation

#### 3.1.2.1 #define SHE_KEY_1 (0x04)

Identifiers for SHE keys.

#### 3.1.2.2 #define SHE_KEY_2 (0x05)

#### 3.1.2.3 #define SHE_KEY_3 (0x06)

#### 3.1.2.4 #define SHE_KEY_4 (0x07)

#### 3.1.2.5 #define SHE_KEY_5 (0x08)

#### 3.1.2.6 #define SHE_KEY_6 (0x09)

**3.1.2.7   #define SHE_KEY_7 (0x0a)**

**3.1.2.8   #define SHE_KEY_8 (0x0b)**

**3.1.2.9   #define SHE_KEY_9 (0x0c)**

**3.1.2.10   #define SHE_KEY_10 (0x0d)**

**3.1.2.11   #define SHE_RAM_KEY (0x0e)**

**3.1.2.12   #define SHE_KEY_DEFAULT (0x00)**

Identifiers for SHE keys extensions.

no key extension: keys from 0 to 10 as defined in SHE specification.

**3.1.2.13   #define SHE_KEY_N_EXT_1 (0x10)**

keys 11 to 20.

**3.1.2.14   #define SHE_KEY_N_EXT_2 (0x20)**

keys 21 to 30.

**3.1.2.15   #define SHE_KEY_N_EXT_3 (0x30)**

keys 31 to 40.

**3.1.2.16   #define SHE_KEY_N_EXT_4 (0x40)**

keys 41 to 50.

**3.1.2.17   #define SHE_STORAGE_CREATE_SUCCESS 0u**

New storage created succesfully.

**3.1.2.18   #define SHE_STORAGE_CREATE_WARNING 1u**

New storage created but its usage is restricted to a limited security state of the chip.

**3.1.2.19   #define SHE_STORAGE_CREATE_UNAUTHORIZED 2u**

Creation of the storage is not authorized.

**3.1.2.20   #define SHE_STORAGE_CREATE_FAIL 3u**

Creation of the storage failed for any other reason.

**3.1.2.21 #define SHE_STORAGE_NUMBER_UPDATES_DEFAULT 300u**

default number of maximum number of updated for SHE storage.

**3.1.2.22 #define SHE_MAC_SIZE 16u**

size of the MAC generated is 128bits.

**3.1.2.23 #define SHE_MAC_VERIFICATION_SUCCESS 0u**

indication of mac verification success

**3.1.2.24 #define SHE_MAC_VERIFICATION_FAILED 1u**

indication of mac verification failure

**3.1.2.25 #define SHE_AES_BLOCK_SIZE_128 16u**

size in bytes of a 128bits CBC bloc

**3.1.2.26 #define SHE_KEY_SIZE 16u /∗∗ SHE keys are 128 bits (16 bytes) long. ∗/**

**3.1.2.27 #define SHE_ENTROPY_SIZE 16u**

**3.1.2.28 #define SHE_RND_SIZE 16u**

**3.1.2.29 #define SHE_CHALLENGE_SIZE 16u /∗ 128 bits ∗/**

**3.1.2.30 #define SHE_ID_SIZE 15u /∗ 120 bits ∗/**

**3.1.3 Enumeration Type Documentation**

**3.1.3.1 enum she_err_t**

Error codes returned by SHE functions.

**Enumerator**

    ***ERC_NO_ERROR*** Success.

    ***ERC_SEQUENCE_ERROR*** Invalid sequence of commands.

    ***ERC_KEY_NOT_AVAILABLE*** Key is locked.

    ***ERC_KEY_INVALID*** Key not allowed for the given operation.

    ***ERC_KEY_EMPTY*** Key has not beed initialized yet.

    ***ERC_NO_SECURE_BOOT*** Conditions for a secure boot process are not met.

    ***ERC_KEY_WRITE_PROTECTED*** Memory slot for this key has been write-protected.

    ***ERC_KEY_UPDATE_ERROR*** Key update did not succeed due to errors in verification of the messages.

    ***ERC_RNG_SEED*** The seed has not been initialized.

    ***ERC_NO_DEBUGGING*** Internal debugging is not possible.

    ***ERC_BUSY*** A function of SHE is called while another function is still processing.

    ***ERC_MEMORY_FAILURE*** Memory error (e.g. flipped bits)

    ***ERC_GENERAL_ERROR*** Error not covered by other codes occured.

**3.1.4 Function Documentation**

**3.1.4.1 uint32_t she_storage_create ( uint32_t *key_storage_identifier,* uint32_t *password,* uint16_t *max_updates_number,* uint8_t ∗ *signed_message,* uint32_t *msg_len* )**

Creates an empty SHE storage.

Must be called at least once on every device before using any other SHE API. A signed message can be provided to authorize the operation. This message is not necessary under some conditions related to chip's lifecycle.

Note that the signed message is not yet supported. should be forced to NULL.

**Parameters**

| *key_storage_identifier* | key store identifier |
|---|---|
| *password* | new password to be associated with the created key store |
| *max_updates_number* | maximum number of updates authorized on this new storage. Cannot be higher than 300. |
| *signed_message* | pointer to a signed message authorizing the operation (NULL if no signed message to be used) |
| *msg_len* | length in bytes of the signed message |

**Returns**

error code

**3.1.4.2 struct she_hdl_s∗ she_open_session ( uint32_t *key_storage_identifier,* uint32_t *password,* void(∗)(void ∗priv, she_err_t err) *async_cb,* void ∗ *priv* )**

Initiate a SHE session. The returned session handle pointer is typed with the struct "she_hdl_s". The user doesn't need to know or to access the fields of this struct. It only needs to store this pointer and pass it to every calls to other APIs within the same SHE session.

Note that asynchronous API is currently not supported. async_cb and priv pointers must be set to NULL.

**Parameters**

| *key_storage_identifier* | key store identifier |
|---|---|
| *password* | password for accesing the key storage |
| *async_cb* | user callback to be called on completion of a SHE operation |
| *priv* | user pointer to be passed to the callback |

**Returns**

pointer to the session handle.

**3.1.4.3 void she_close_session ( struct she_hdl_s ∗ *hdl* )**

Terminate a previously opened SHE session

**Parameters**

| hdl | pointer to the session handler to be closed. |
|-----|----------------------------------------------|

**3.1.4.4  she_err_t she_cmd_generate_mac ( struct she_hdl_s ∗ *hdl,* uint8_t *key_ext,* uint8_t *key_id,* uint16_t *message_length,* uint8_t ∗ *message,* uint8_t ∗ *mac* )**

Generates a MAC of a given message with the help of a key identified by key_id.

**Parameters**

| hdl | pointer to the SHE session handler |
|-----|-------------------------------------|
| key_ext | identifier of the key extension to be used for the operation |
| key_id | identifier of the key to be used for the operation |
| message_length | lenght in bytes of the input message |
| message | pointer to the message to be processed |
| mac | pointer to where the output MAC should be written (128bits should be allocated there) |

**Returns**

error code

**3.1.4.5  she_err_t she_cmd_verify_mac ( struct she_hdl_s ∗ *hdl,* uint8_t *key_ext,* uint8_t *key_id,* uint16_t *message_length,* uint8_t ∗ *message,* uint8_t ∗ *mac,* uint8_t *mac_length,* uint8_t ∗ *verification_status* )**

Verifies the MAC of a given message with the help of a key identified by key_id.

**Parameters**

| hdl | pointer to the SHE session handler |
|-----|-------------------------------------|
| key_ext | identifier of the key extension to be used for the operation |
| key_id | identifier of the key to be used for the operation |
| message_length | lenght in bytes of the input message |
| message | pointer to the message to be processed |
| mac | pointer to the MAC to be compared (implicitely 128 bits) |
| mac_length | number of bytes to compare (must be at least 4) |
| verification_status | pointer to where write the result of the MAC comparison |

**Returns**

error code

**3.1.4.6  she_err_t she_cmd_enc_cbc ( struct she_hdl_s ∗ *hdl,* uint8_t *key_ext,* uint8_t *key_id,* uint32_t *data_length,* uint8_t ∗ *iv,* uint8_t ∗ *plaintext,* uint8_t ∗ *ciphertext* )**

CBC encryption of a given plaintext with the key identified by key_id.

**Parameters**

| hdl | pointer to the SHE session handler |
| --- | --- |
| key_ext | identifier of the key extension to be used for the operation |
| key_id | identifier of the key to be used for the operation |
| data_length | lenght in bytes of the plaintext and the cyphertext. Must be a multiple of 128bits. |
| iv | pointer to the 128bits IV to use for the encryption. |
| plaintext | pointer to the message to be encrypted. |
| ciphertext | pointer to ciphertext output area. |

**Returns**

error code

**3.1.4.7 she_err_t she_cmd_dec_cbc ( struct she_hdl_s ∗ *hdl,* uint8_t *key_ext,* uint8_t *key_id,* uint32_t *data_length,* uint8_t ∗ *iv,* uint8_t ∗ *ciphertext,* uint8_t ∗ *plaintext* )**

CBC decryption of a given ciphertext with the key identified by key_id.

**Parameters**

| hdl | pointer to the SHE session handler |
| --- | --- |
| key_ext | identifier of the key extension to be used for the operation |
| key_id | identifier of the key to be used for the operation |
| data_length | lenght in bytes of the plaintext and the cyphertext. Must be a multiple of 128bits. |
| iv | pointer to the 128bits IV to use for the decryption. |
| ciphertext | pointer to ciphertext to be decrypted. |
| plaintext | pointer to the plaintext output area. |

**Returns**

error code

**3.1.4.8 she_err_t she_cmd_enc_ecb ( struct she_hdl_s ∗ *hdl,* uint8_t *key_ext,* uint8_t *key_id,* uint8_t ∗ *plaintext,* uint8_t ∗ *ciphertext* )**

ECB encryption of a given plaintext with the key identified by key_id.

**Parameters**

| hdl | pointer to the SHE session handler |
| --- | --- |
| key_ext | identifier of the key extension to be used for the operation |
| key_id | identifier of the key to be used for the operation |
| plaintext | pointer to the 128bits message to be encrypted. |
| ciphertext | pointer to ciphertext output area (128bits). |

**Returns**

error code

**3.1.4.9   she_err_t she_cmd_dec_ecb ( struct she_hdl_s** ∗ *hdl,* **uint8_t** *key_ext,* **uint8_t** *key_id,* **uint8_t** ∗ *ciphertext,* **uint8_t** ∗ *plaintext* **)**

ECB decryption of a given ciphertext with the key identified by key_id.

**Parameters**

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |
| *key_ext* | identifier of the key extension to be used for the operation |
| *key_id* | identifier of the key to be used for the operation |
| *ciphertext* | pointer to 128bits ciphertext to be decrypted. |
| *plaintext* | pointer to the plaintext output area (128bits). |

**Returns**

error code

**3.1.4.10   she_err_t she_cmd_load_key ( struct she_hdl_s** ∗ *hdl,* **uint8_t** *key_ext,* **uint8_t** *key_id,* **uint8_t** ∗ *m1,* **uint8_t** ∗ *m2,* **uint8_t** ∗ *m3,* **uint8_t** ∗ *m4,* **uint8_t** ∗ *m5* **)**

Update an internal key of SHE with the protocol specified by SHE.

**Parameters**

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |
| *key_ext* | identifier of the key extension to be used for the operation |
| *key_id* | identifier of the key to be used for the operation |
| *m1* | pointer to M1 message - 128 bits |
| *m2* | pointer to M2 message - 256 bits |
| *m3* | pointer to M3 message - 128 bits |
| *m4* | pointer to the output address for M4 message - 256 bits |
| *m5* | pointer to the output address for M5 message - 128 bits |

**Returns**

error code

**3.1.4.11   she_err_t she_cmd_load_plain_key ( struct she_hdl_s** ∗ *hdl,* **uint8_t** ∗ *key* **)**

Load a key as plaintext to the RAM_KEY slot without encryption and verification.

**Parameters**

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |
| *key* | pointer to the plaintext key to be loaded - 128bits |

**Returns**

> error code

**3.1.4.12 she_err_t she_cmd_export_ram_key ( struct she_hdl_s ∗ *hdl,* uint8_t ∗ *m1,* uint8_t ∗ *m2,* uint8_t ∗ *m3,* uint8_t ∗ *m4,* uint8_t ∗ *m5* )**

exports the RAM_KEY into a format protected by SECRET_KEY.

**Parameters**

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |
| *m1* | pointer to the output address for M1 message - 128 bits |
| *m2* | pointer to the output address for M2 message - 256 bits |
| *m3* | pointer to the output address for M3 message - 128 bits |
| *m4* | pointer to the output address for M4 message - 256 bits |
| *m5* | pointer to the output address for M5 message - 128 bits |

**Returns**

> error code

**3.1.4.13 she_err_t she_cmd_init_rng ( struct she_hdl_s ∗ *hdl* )**

initializes the seed and derives a key for the PRNG. The function must be called before CMD_RND after every power cycle/reset.

**Parameters**

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |

**Returns**

> error code

**3.1.4.14 she_err_t she_cmd_extend_seed ( struct she_hdl_s ∗ *hdl,* uint8_t ∗ *entropy* )**

extends the seed of the PRNG by compressing the former seed value and the supplied entropy into a new seed which will be used to generate the following random numbers. The random number generator has to be initialized by CMD_INIT_RNG before the seed can be extended.

**Parameters**

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |
| *entropy* | pointer to the entropy vector (128bits) to use for the operation |

**Returns**

> error code

**3.1.4.15 she_err_t she_cmd_rnd ( struct she_hdl_s ∗ *hdl,* uint8_t ∗ *rnd* )**

returns a vector of 128 random bits. The random number generator has to be initialized by CMD_INIT_RNG before random numbers can be supplied.

**Parameters**

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |
| *rnd* | pointer to the output address for the generated 128bits random vector |

**Returns**

> error code

**3.1.4.16 she_err_t she_cmd_get_status ( struct she_hdl_s ∗ *hdl,* uint8_t ∗ *sreg* )**

returns the content of the status register

**Parameters**

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |
| *sreg* | pointer to the output address for status register(8bits) |

**Returns**

> error code

**3.1.4.17 she_err_t she_cmd_get_id ( struct she_hdl_s ∗ *hdl,* uint8_t ∗ *challenge,* uint8_t ∗ *id,* uint8_t ∗ *sreg,* uint8_t ∗ *mac* )**

returns the identity (UID) and the value of the status register protected by a MAC over a challenge and the data.

**Parameters**

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |
| *challenge* | pointer to the challenge vector (128bits) |
| *id* | pointer to the output address for the identity (120bits) |
| *sreg* | pointer to the output address for status register(8bits) |
| *mac* | pointer to the output address for the computed MAC (128bits) |

**Returns**

> error code

**3.1.4.18 she_err_t she_cmd_cancel ( struct she_hdl_s ∗ *hdl* )**

interrupt any given function and discard all calculations and results.

**Parameters**

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |

**Returns**

error code

| | |
|---|---|
| *hdl* | pointer to the SHE session handler |

## 3.2 She_storage

SHE NVM storage API.

**Functions**

- struct she_storage_context ∗ she_storage_init (void)
- int32_t she_storage_terminate (struct she_storage_context ∗nvm_ctx)

### 3.2.1 Detailed Description

SHE NVM storage API.

### 3.2.2 Function Documentation

#### 3.2.2.1 struct she_storage_context∗ she_storage_init ( void )

Initialize SHE storage manager.

**Returns**

pointer to the storage context

#### 3.2.2.2 int32_t she_storage_terminate ( struct she_storage_context ∗ *nvm_ctx* )

terminates the SHE storage manager.

**Parameters**

| *ctx* | pointer to the context of the storage manager to be closed. |
|---|---|

**Returns**

0 on success. other value on failure.

# Index