NXP HSE BASIC FIRMWARE FREQUENTLY ASKED QUESTIONS

Table of Contents

HSE FW ATTRIBUTES

Q: How do I configure/enable more Message Units (MUs)?

By default, only MU0 is enabled and it can't be disabled. To enable more MUs, the *hseSetAttrSrv_t* service can be used with the *HSE_MU_CONFIG_ATTR_ID* attribute ID. For more details, refer to HSE Demo Application and the following structure definition from hse interface: *hseAttrMUConfig_t.*

Q: How can I partition the application domain RAM to be accessible only to certain MU instances?

By default, HSE restricts the address space for where the data for service requests can be provided to its own address space and other reserved areas, however independent of MU.

To partition the host RAM space per MU instance (i.e. obtain isolation between applications), hseSetAttrSrv_t service can be used to allocate for each MU instance the allowed list of address ranges from which data can be provided to HSE.

For more details, refer to hseAttrAllMuMemRegions_t, hseAttrMemRegion_t and hseAttrMuMemRegions_t structures in HSE interface and API RM.

Q: How do I configure the ADK/P (debug key or password)?

The debug key/password (ADK/P) is a 128-bit value than can be configured using *HSE_APP_DEBUG_KEY_ATTR_ID* attribute. ADK/P can be written only once (UTEST attribute), and the operation is allowed only in CUST_DEL Life Cycle. By default, application debug authorization mode is password based. To enable the challenge-response debug authorization mode (based on the key), the application shall use the *HSE_DEBUG_AUTH_MODE_ATTR_ID* attribute. For more details, refer to HSE Demo Application and the following structure definition from hse interface and API RM: *hseAttrApplDebugKey_t, hseAttrDebugAuthMode_t.*

Q: What does ADKm mean? How can I configure it?

The Application debug key/ password (ADK/P attribute) can be diversified with UID before being written in UTEST area. The 128-bit value set with *HSE_APP_DEBUG_KEY_ATTR_ID* attribute will be interpreted as ADKPm (customer's master key/ password). The following derivation is used:

- hUID = SHA2_256(UID)
- hADKPm = SHA2_256(ADKPm)
- ADKP {for debugger} = AES256-ECB(hUID(16 bytes..0 to 15)), where the key is hADKPm; {ADKPm = customer's master key/ password}

The following attributes must be set to enable ADKPm usage:

- Set *hseAttrExtendCustSecurityPolicy_t* attribute (enableADKm is enabled)
- Set *hseAttrApplDebugKey_t* attribute.

Note that *hseAttrExtendCustSecurityPolicy_t* security policy must be set before *hseAttrApplDebugKey_t* attribute, and these attributes can be set only once (UTEST attributes)

Q: How can I know whether application password has been programmed successfully?

User can initiate the GetAttribute service for reading the ADKP attribute. HSE FW in response to this service will return the SHA224(ADKP) to host application. User should also calculate the SHA 224 HASH of ADKP at its end to verify that HSE FW has correctly programmed the ADKP in UTEST area. The firmware only returns the first 16 Bytes of the SHA224(ADKP).

Q: How do I advance the Life Cycle?

SOC is delivered in CUST_DEL lifecycle. The Lifecycle should be advanced using *HSE_SECURE_LIFECYCLE_ATTR_ID* (see *hseAttrSecureLifecycle_t*). The Life Cycle attribute can be set only once (UTEST attribute) and only if the ADK/P is already set in UTEST. Lifecycle can be advanced to OEM_PROD and IN_FIELD lifecycle. For more details, refer to HSE Demo Application and attribute definition from the HSE interface.

Q: What attributes are stored in UTEST area?

The attributes from the categories OTP-ATTR or OTP-ADVANCE-ATTR are stored in UTEST area. This means that they can only be set once, advanced respectively, in the UTEST area.
One can find more information on the attributes classification in the HSE interface.

HSE FW administrative and installation

Q: What is the firmware feature flag? How do I enable it?

K3xx family devices are delivered with only secure BAF programmed. Customer has the option to configure the device with secure firmware feature enabled or disabled. By default, it is assumed that HSE FW will not be used by the user hence all the resources (HSE Code flash area, HSE data flash area) required by HSE FW are free for application to use. But if customer wants to use HSE FW feature on the device then customer has to program the location 0x1B000000 with 8 bytes of random value and give a reset. After the reset, secure BAF will reserve the 176KB area of code flash and 168KB of data flash area for HSE area.
For some custom firmware versions, this step might not be required. Please refer to annexure document of custom firmware for more details.

Q: How do I install HSE FW on a virgin device without secure boot?

Firmware feature flag must be enabled before installing the firmware. HSE FW can be installed in the system using three methods:

- **Method 1**: Program the encrypted image of HSE FW at start location of code flash area i.e. 0x00400000 and give a reset. SBAF installs the HSE FW after reset.
- **Method 2**: Program the address encrypted image of HSE FW in IVT and program the encrypted HSE FW image at the provided address. After programming, provide a reset.
- **Method 3**: Installing the HSE FW through MU interface. Refer to HSE FW reference manual for more details. The advantage of this approach is that user doesn't need to program the encrypted image in flash. It can be saved in RAM also.

HSE FW installation status can be checked by Application by polling the bit number 24 at FSR register in MU at address 0x4038C108
For more details, refer to HSE Firmware Reference Manual.

Q: How do I update the HSE FW image?

Program the HSE FW encrypted image in host code flash area or ram area and issue a firmware update request to HSE firmware with the address where new image is programmed. The firmware update operation can be performed in one shot or in streaming mode. For more details refer to the structure *hseFirmwareUpdateSrv_t* in *hse_interface.h* file.

Q: What is the advantage of using the streaming mode in firmware update command?

The advantages of streaming mode is that user can perform the firmware update operation without the need to store the full image i.e. 128KB size in RAM or flash area. By issuing multiple firmware update command with chunk size multiple of 64 bytes, user can perform firmware update operation. For smaller memory footprint devices like S32K3x2 and S32K3x1 variants, this feature can be very useful.

Q: What do the term pink firmware image mean?

The HSE Firmware pink image is the encrypted and authenticated image provided by NXP and encrypted with NXP keys.

Q: What is SYS-Authorization and how can I use it?

The execution of certain HSE services is conditioned by the execution rights granted to the host:

- SuperUser (OEM or CUST) rights - high execution privileges and no limitation on service requests.
- User rights – restricted execution privileges

After reset, the SYS rights are synchronized with Life cycle (LC):

- if the Life Cycle is CUST_DEL, CUST SuperUser rights are granted

- if the Life Cycle is OEM_PROD, OEM SuperUser rights are granted
- if the Life Cycle is IN_FIELD, User rights are granted

During run-time (e.g IN_FIELD Life cycle), the User rights can be temporarily elevated to SuperUser(CUST/OEM) using HSE Authorization Request/Response services. To be able to use these services, the Host Application must have previously provisioned an authorization key, which is an NVM key that has the HSE_KF_USAGE_AUTHORIZATION and HSE_KF_USAGE_VERIFY key flags set.

The authorization procedure is executed as follows:

- The host triggers the authorization request and the HSE returns a random challenge (refer to *hseSysAuthorizationReqSrv_t* service)
- The host then computes a signature over the challenge using the selected authorization key and provides it to the HSE (refer to *hseSysAuthorizationRespSrv_t* service)
- If the response provided is the expected one, the HSE grants the host with SU rights
- Also, the host has the possibility to go to User mode calling *hseSysAuthorizationReqSrv_t* service

The SYS Authorization also depends on security policy attributes, such as *hseAttrExtendCustSecurityPolicy_t* and *hseAttrExtendOemSecurityPolicy_t*. For example, in CUST_DEL lifecycle the host can be forced in User mode if the "startAsUser" field of *hseAttrExtendCustSecurityPolicy_t* attribute is set.

NOTE:

- The device can advance the Lifecycle from CUST_DEL directly to IN_FIELD. In this case the OEM_PROD lifecycle is not used, e.g. all the NVM keys are provisioned in CUST_DEL life cycle, including OEM keys.
- If no authorization key is provisioned, it means the Authorization procedure cannot be used.

For more details, refer to HSE Demo App (readme file) and the HSE interface.

Q: What is the use of the "Boot Data Sign" service?

The Boot Data Sign service is used for generating a signature over the following images:

- IVT image
- Application image - when Basic Secure Boot is used

The signature of boot artifacts IVT is verified by BAF when(IVT_AUTH is set in UTEST area) and the Application Image signature is verified by HSE FW when Basic Secure Boot is used in case of BOOT_SEQ=1. To generate a signature over one of the above images, the *hseBootDataImageSignSrv_t* service must be used. To enable IVT/DCD/ST authentication, the application must set the *hseAttrConfigBootAuth_t* attribute. It should be ensured that latest version of SBAF and HSE Firmware should be present in the device to use this service.

Q: What is SYS-IMAGE? What does it contain

The SYS-IMG is an HSE FW artifact which contains the user configuration. It consists of the following set of information:

- NVM attributes

- SMR and CR configuration
- KEY configuration – catalogs format and keys
- Monotonic counters

If any of the above configuration items are updated, hse programs it in secure data flash area.

Q: How can I use User mode in CUST_DEL life cycle (without changing the life cycle)?

The host can go to User mode calling hseSysAuthorizationReqSrv_t service with sysRights parameter set to HSE_RIGHTS_USER.

Q: Can I simulate the OEM_PROD or IN_FIELD Life Cycle (Super User rights) being in CUST_DEL Life Cycle?

Yes, from HSE FW perspective (only SW, not HW), there are three options:

- Set the Life Cycle to HSE_LC_SIMULATED_OEM_PROD or HSE_LC_SIMULATED_IN_FIELD using the hseAttrSecureLifecycle_t attribute (see attribute description)
- Or use hseSysAuthorizationReqSrv_t with sysRights as HSE_RIGHTS_USER (see service description). After reset, HSE starts with the default System right (in this case CUST Super User rights)
- Or set the extended security policies (startAsUser member of hseAttrExtendCustSecurityPolicy_t or hseAttrExtendOemSecurityPolicy_t structures) such that HSE operates with limited access rights (as IN_FIELD). In this case, after reset, HSE starts User access rights.

HSE FW Secure Boot configuration

Q: When is the boot artifact i.e.IVT checked?

The boot artifacts are check by SBAF. If they are signed (see hseBootDataImageSignSrv_t service), the signature is verified by SBAF only if the hseAttrConfigBootAuth_t attribute is set.

Q: How do I configure secure booting using Secure Memory Region (SMR)?

To configure SMR boot, one needs to install at least one CR entry(*hseCrEntryInstallSrv_t*) and one SMR entry (*hseSmrEntryInstallSrv_t*). The core reset entry needs an SMR entries mask which specifies which SMR entries need to be successfully verified, in order for the core to be successfully booted. For an example as to how to configure secure booting using SMR check the HSE Demo App which has detailed examples.

Q: Can a SMR entry be updated in User mode (IN_FIELD LC)?

YES, under certain conditions:

On SMR installation (see hseSmrEntryInstallSrv_t) only the following parameters can be updated: pSmrSrc, smrSize, pInstAuthTag. Any other parameters (e.g. keyHandle, configFlags etc) cannot be updated in User mode (these parameters can be updated only by having SuperUser rights).

Q: What does Basic Secure Boot (BSB) mean? How can I use it?

Basic Secure Boot (BSB) is a simplified boot scheme supported within hseBootDataImageSignSrv_t service. BSB can boot only one core (the booted core can start other cores).
The application image should contain the header that includes all information needed for BSB (i.e. same as in un-secure boot). The signature should be appended to the end of AppBL and generated using a key derived from ADK/P. The securing process is similar to IVT authentication generation/verification starting from revision 2, only difference being that HSE FW (not SBAF) manages it

Q: Can I update the Core Reset (CR) and SMR in IN_FIELD Life cycle?

To change a Core Reset entry, host needs to have SuperUser access rights (IN_FIELD LC). To gain SuperUser rights, the host must perform the system authorization procedure using an authorization key (installed before). If there is no authorization key installed, CR cannot be changed.
SMR can be updated in IN_FIELD LC without authorization by providing different values only for the following parameters: pSmrSrc, smrSize, pInstAuthTag. Otherwise, host needs SuperUser rights (needs to perform system authorization).

Q: Can I boot encrypted application images securely?

Yes. SMR supports an encryption scheme such confidentiality is also provided for the secure memory region. The encryption can be carried out in two ways:

- Using AEAD-GCM with null AAD. In this scheme, the generated GMAC tag over the encrypted image must also be provided with the SMR.
- Using AES-CTR. In this case HSE will generate at installation time the authenticity over the encrypted image. The pGmacTag field is not used.

The encrypted SMR is a generic mechanism and works for any memory region that is loaded (pSmrDest address is provided), independent of the scope (i.e. not only for boot images).
For more details, checkout hseSmrDecrypt_t structure and its usage within hseSmrEntry_t in the HSE interface, along with SMR chapter in HSE RM.

Q: Is there a way to improve the secure boot performance on default clock?

Yes. Default Secure Boot is executed by HSE Core on 48MHz. To enable the PLL configuration, Host must enable the PLL enable bit in BCW in the IVT. HSE Firmware enables the PLL in two modes, Option A (only on supported devices) and Option B. For more details on these clocking options, refer to SoC Reference Manual, Clocking Chapter. As a prerequisite, host is expected to enable FXOSC via SBAF by configuring the UTEST at location 0x1B000050. For more details on FXOSC configuration by SBAF, refer to SoC Reference Manual, Boot Chapter.

Option B can only be enabled if the gasket is configured by the user via DCF record. For more details on gasket configuration, refer to SoC Reference Manual, DCF Record Sheet. If the DCF record is not programmed and all the other configurations are valid, device will function with Option A (if it is supported on the device, else it will function on default clock).

Q: What steps to be followed to make sure that secure boot works seamlessly when application image is updated?

User must make sure that all SMR regions must be updated if there is a change in memory content because of application update. Super user rights must be attained before updating the SMR region. To ensure that SMR regions have been updated correctly, user can optionally perform the on demand verification of all SMR regions before issuing a reset. In case SMR0 is used with BOOT_MAC_KEY handle i.e. secure boot needs to be followed as per SHE specification, BOOT_MAC calculated for new application image also needs to be updated using SHE_LOAD_KEY service. The above steps remain same for FullMem and ABSwap configuration. However, for AB swap configuration, in case of on demand SMR verification, one additional steps needs to be done which is to pass the information related to address translation logic. For more details, refer to HSE firmware reference manual.

HSE FW Keys

Q: How do I format the key catalog?

The Key Catalog formatting is done using a service called hseFormatKeyCatalogsSrv_t that allows the host to configure key groups of certain types. These configured key slots can be provisioned by the host using the import key, key generation or key derivation (from other keys) services. Two types of catalogs can be formatted:

- NVM key catalog (persistent keys)
- RAM key catalog (ephemeral keys which are not persistent – wiped at device reset)

For each key group with Key Catalog, the following information have to be provided:

- MU instance(s); the key can be accessed only using the defined MU(s) in the group.
- Key type (e.g. AES key group, ECC public key group)
- Number of key slots
- The maximum size of the key in bits (e.g. 128-bit for AES128, 256-bit for ECC p256 curve)

For more details, refer to HSE Demo App (readme file) and hse interface.

Q: How do I configure and install keys?

Each key has a value and certain proprieties (called *hseKeyInfo_t*) contained into the key slot. At installation the value and the key properties must be provided. The host can dynamically configure the keys using one of the following services:

- Key generation (symmetric and asymmetric keys can be generated); refer to *hseKeyGenerateSrv_t*.
- Import key; refer to hseImportKeySrv_t service
- Key derivation (from other keys); refer to *hseKeyDeriveSrv_t* and *hseKeyDeriveCopyKeySrv_t* services.

The usage of the above services may require SuperUser rights, depending on which key slot is targeted (e.g. NVM or RAM key).
For more details, refer to HSE Demo App (readme file) and hse interface.

Q: How can I generate keys?

The symmetric or asymmetric keys can be generated using the hseKeyGenerateSrv_t service to NVM or RAM key slots. When the service is called, different restrictions are applied based on SuperUser/User rights:

- SuperUser key restrictions:
  - NVM keys can only be generated in empty slots (if you need to overwrite a key, erase the previous key first)
  - RAM keys can always be generated (RAM keys can be overwritten)
- User key restrictions:
  - NVM keys can NOT be generated.
  - RAM keys can always be generated (RAM keys can be overwritten)

.

Q: How are the keys used with a crypto operation?

All keys used with crypto operation are referenced by a unique Key Handle. The Key Handle is defined as a 32-bit value as follow:
keyHandle = 0x00(byte3) || key catalog id(byte2) || key group index(byte1) || keys slot index(byte0), where

- key catalog id identifies the key catalog: ROM, NVM, RAM
- key group index identified the index of group defined in the catalog
- keys slot index is between 0 and (p-1), where p is the maximum number of keys defined in the group.

Q: Can I avoid using OEM keys (including OEM Life Cycle)?

YES. This means that all NVM keys are provisioned in CUST_DEL lifecycle. In this case, the life cycle will be advanced from CUST_DEL directly to IN_FIELD

Q: How can I generate the TLS session keys using HSE services?

To generate the TLS session keys for a session using ephemeral (elliptic curve) Diffie-Hellman key negotiation, the application must follow the steps below:

1. Shared secret computation:
    a. generate a pair of ephemeral keys using hseKeyGenerateSrv_t service (e.g. generate an ECC key pair exporting the public key to the host).
    b. sign the server_key_exchange/client_key_exchange message with the key corresponding to the authentication certificate of the device
    c. import the other party's certificate public key from the certificate message
    d. import the other party's Diffie-Hellman public key from the client_key_exchange/server_key_exchange message using hseImportKeySrv_t. The key signature must be verified using the other party's certificate public key (the key container will consist of the entire key_exchange message).
    e. compute the Diffie-Hellman shared secret (in a SHARED_SECRET slot) using hseDHComputeSharedSecretSrv_t service.
2. Derive the TLS keys:
    a. derive the TLS the master secret, the key block and the verify data using the hseKdfTLS12PrfScheme_t scheme from key derivation service (hseKeyDeriveSrv_t). This service must be call multiple time to generate:
        i. the master secret from pre-master secret (the shared secret compute above)
        ii. the TLS key_block from master (for TLS key expansion step)
        iii. client/server verify_data for TLS Finished messages

The key material (key_block) is generated in a SHARED_SECRET slot which is used as temporary slot. To be able to use the generated keys with crypto operations, the application has to use hseKeyDeriveCopyKeySrv_t service to copy (one-by-one) each MAC and encryption keys to RAM key slots of a certain key type (e.s. AES key slots).
Note: Client authentication is mandatory for any TLS session supported by HSE

Q: How do I install a public certificate stored by application in flash?

The import key service (*hseImportKeySrv_t*) can be used to import a public key certificate which is stored in plain in application flash area. The key types *_PUB_EXT have to be used as a destination key slot to import such a certificate (e.g. *HSE_KEY_TYPE_ECC_PUB_EXT*, *HSE_KEY_TYPE_RSA_PUB_EXT*). For these key types, HSE stores only the key properties and the pointers to the public key values and certificate, as well as an internal authentication tag calculated over the certificate. This authentication tag is verified by the HSE firmware whenever the related key is used by the host.
To import a PUB_EXT key, the host shall provide a pointer to that certificate, pointer(s) to key value(s) within the certificate and pointer to the signature (computed over the certificate) along with the signature key handle.
For more details refer to *hseKeyType_t* and *hseImportKeySrv_t* definitions.

Q: When the device is in IN_FIELD life cycle, can both OEM and CUST keys be used?

YES. Keys are bound to one or more MU instances. The MU Instance assigned to the key (defined within Key Catalog Format) shall matching the MU Instance on which the crypto service was sent. Also, each key as a set of flags that specifies the operations and restrictions that can be apply on key, such as: sign flag, verify flag, encrypt flag, decrypt flag etc. A key that has only the sign flag set cannot be used for verification.
For more details refer to *hseKeyFlags_t, hseKeyInfo_t* and *hseFormatKeyCatalogsSrv_t* services

Q: How can I improve the overall time to load the multiple keys in sys image?

HSE FW will always programs the keys in data flash on every import key request.
Programming of keys in data flash involves erasing and programming on passive sector.
Because of above steps, the time to load a single key is typically ~55 to 60 msec. To optimize this time, user can do following steps:

- Set the attribute hsePublishNvmKeystoreRamtToFlash_t
- Load and update all the required keys using various key management services. In this step, HSE FW will not load the keys in flash
- Publish all keys in data flash using the service HSE_SRV_ID_PUBLISH_NVM_KEYSTORE_RAM_TO_FLASH

The above steps will reduce the average time to load a single key significantly. For example, the time to load the 50 keys individually would have taken 50*60 msec = 3 seconds. But with above optimization the time will reduce to 60 msec. This means that average time to load a single will be ~1.5 msec

Q: What is the use of NXP ROM keys?

ROM key catalog; keys stored in secure NVM, provisioned by NXP before shipment. This catalog contains secret keys, as well as RSA and ECC public keys, controlled and managed by NXP. These keys can be leveraged as root of trust to e.g. provision customer keys in a secure fashion. For more details, please contact NXP support team.

Q: How to verify sanity of key loaded in HSE firmware?

In case user wants to verify the correctness of the key loaded in HSE firmware, it can be done using the service “HSE_SRV_ID_KEY_VERIFY”. This service is used to verify a CMAC, SHA256, SHA384 or SHA512 over   a key stored inside HSE. The CMAC, SHA256 or SHA384 are provided by the application.

HSE FW Crypto/Utils

Q: What are the mechanisms to interact with HSE?

Depending on the type of SoC a total of 2 HSE interface IDs (Message Unit instances) can be used. Each interface ID has up to 4 channel IDS (set of transmit and receive registers on both sides of MU). For sending data or messages from one MU-side to the other MU-side, each MU provides up to 4 transmit registers and up to 4 receive registers on each side of the MU.
The following communication mechanism are used:

1) Service Request/Response over channels. Service requests for HSE Services are submitted to HSE Host Interfaces. The requests are queued internally by HSE firmware and assigned to the right accelerator (symmetric or asymmetric). A status response is signal to the host after the service execution. For more details, refer to hseSrvDescriptor_t and hseSrvResponse_t from interface.
2) HSE Status & Channel Status (at MU level) that can be read by the Application:
    - LSB 4bits are allocated for channels (idle or busy channel)
    - MSB 16bits are allocated for hse status: init OK, RNG init OK, Boot OK etc

   For more details, refer to hseStatus_t from interface.
3) Asynchronous notification errors (at MU level) signaled by HSE to HOST through interrupts. Each time a bit is set (an error is triggered) an interrupt occurs at host size. For example, if a FATAL ERROR is triggered, HSE FW will shut down the MU communication. In this case the application should perform a reset. For more details, refer to hseError_t from interface.

Q: What RNG class should I use?

Depending on the application requirements, the following RNG classes can be used:

- DRG.3 class: prediction resistance disabled; more efficient in terms of performance
- DRG.4: prediction resistance enabled (impact the performance); get random service invokes RNG reseed on each call.
- PTG.3: maximum prediction resistance; it will reseed for each 16 bytes of random data requested.

For more details refer to *hseGetRandomNumSrv_t* service.

Q: Can HSE_GET_RANDOM_NUM_SRV request fail? What happens if the Get Random Request is called again?

The RNG can fail due to temperature and voltage variation. If the RNG fails operating, the HSE_STATUS_RNG_INIT_OK status bit will be cleared (see hseStatus_t). If the Get Random service is called again, HSE firmware will re-kick the RNG initialization before requesting the random number.

HSE FW Others

Q: What is recovery mode and why the SOC goes into this mode?

Recovery mode is initiated by SBAF and HSE FW when there is no valid IVT configured or application can't be booted either by SBAF or HSE FW. In this case HSE core enables both application core at default boot location of 0x2040012C. At this location while(1) code is placed so that application can attach the debugger to its core. After attaching the debugger, application can do flash their application and valid IVT on the system so that from next reset, application is booted at the desired address.

Q: For each life cycle, what will be the state of the application debugger?

The application debugger has the following states:

- In CUST_DEL lifecycle, the debugger is OPEN.
- In OEM_PROD lifecycle, the debugger is CLOSED (password based, or challenge-request based).
- In IN_FIELD lifecycle, the debugger is CLOSED (password based, or challenge-request based).
- In PreFA lifecycle, the debugger is CLOSED. (password based, or challenge-request based).

Q: How can I know what algorithms are supported by the firmware?

The user can check the configuration macros from the *hse_b_config.h* header file. The host can also get the *HSE_CAPABILITIES_ATTR_ID* attribute which returns a 64 bitmap, each bit being assigned for a supported HSE feature.
For more details, refer to *hseAttrCapabilities_t*.

Q: What does HSE_ERR_GENERAL error mean?

This error signals an internal fatal error the HSE sub-system terminates MU communication and shuts down. The HSE will not operate any longer after such an error. The only way to recover from this state is by resetting the system.
The HSE FW signals multiple fatal errors and warnings, for more in depth descriptions on the scenarios and sources of each of the errors, refer to HSE Interface (hseError_t).

Q: How to enable the feature of AB Swap on K3 family?

There are two options to enable AB Swap feature:

- Installing the AB swap firmware, Check the first byte of the encrypted image, "0xDB" denotes the encrypted image for AB swap configuration. The SBAF will install the HSE firmware in the Active partition and sets the Flash configuration to AB swap. When the Flash configuration is set to AB swap, it is not possible to go back to the Full Mem configuration. Hence, only an AB swap HSE firmware image can be used in this configuration.
- Updating the AB swap firmware, The Full Mem firmware must be installed in the system. The Application must call the firmware update service with the address of encrypted AB swap image. The Host must ensure that valid Application is present in the Flash memory along with IVT, On the next reset, the new AB swap HSE firmware is executed and boot up the Application.

Application must ensure that valid application is present in block 0 and block 1 along with IVT so that after the reset that application can be booted by HSE firmware

Q: How much area of code flash does application should free for HSE firmware usage?

Application can determine the size of HSE firmware by the size of encrypted HSE firmware image. Whatever is the size of encrypted HSE FW image, application should subtract the 4K from that and should reserve that amount of space in code flash and in data flash.

If HSE FW type is OTA disabled:

HSE reserved area in code flash block 3 will be	: 48KB + Size of HSE firmware
HSE reserved area in data flash will be	: maximum of (40KB + Size of HSE firmware) or 128KB

For standard firmware type, the size of Full Mem HSE FW will be 128KB.

If HSE FW type is OTA enabled:

HSE reserved area in code flash block 1 will be	: Size of HSE firmware
HSE reserved area in code flash block 3 will be	: Size of HSE firmware
HSE reserved area in data flash will be	: 128KB

For standard firmware type, the size of AB Swap HSE FW will be 176KB.

Q: How will I get to know whether I have to update the SBAF in a device?

If the version of SBAF present in the system is not compatible with the HSE FW version present then SBAF must be updated using HSE_SRV_ID_SBAF_UPDATE service. Please refer to chapter 14 in HSE FW reference manual for more details.

Q: How to ensure the flash synchronization between application and HSE core to avoid read while write exception?

HSE Firmware blocks program and erase operation of a flash block while reading and during program and erase by HSE core, App core cannot read from the same flash block due to read-while-write issues. Application should check the status of Read Lock Bits in CONFIG_REG3 (0x4039C028) before reading the block and should also check the status of Write Lock Bits in CONFIG_REG3 before programing or erasing the same flash block. Refer to HSE FW reference manual, Chapter 14 for more details on flash synchronizations and register descriptions.

Q: What are the differences between Standard and Premium HSE FW packages?

Premium HSE FW packages are custom firmware releases which contains customer specific requirements. Please contact your NXP sales representative for more details.

Q: What is the reason of incompatibility between older version of SBAF and latest version of HSE Firmware? How do I make SBAF and HSE FW version compatible?

The reason of incompatibility is because authentication scheme is improved in which random IV is used to calculate the GMAC of security artifacts like IVT, AppBL and secure recovery images. Previously it was fixed IV value. Because of which artifacts signed by HSE FW using BOOT_DATA_SIGN service can't be verified by SBAF. Hence it is recommended to use latest version of SBAF to use all features of HSE FW without any limitations.

Q: Can the application running on host side do the flash programming at the time of giving the service to HSE Firmware?

Although it is not recommended that application should be performing program or erase operation on any flash block because it might interfere with HSE FW operation in case any of address passed in any of the parameter lies in the flash block on which high voltage operation is in progress. In that case, HSE FW may become non operational because of read while write exception. However, it is mandatory user must not be doing any flash programming on flash block from where HSE FW is executing its code. This flash block will vary depending on device variant of S32K3 device or FullMem/AB swap configuration.

Q: How to make sure that device contains latest version of SBAF and HSE firmware?

SBAF version information can be read from the address location 0x4039C020.
The latest version of HSE FW can be read by reading the version number attribute "HSE_FW_VERSION_ATTR_ID" by giving the service ID "HSE_SRV_ID_GET_ATTR".
The value of latest version of SBAF and HSE FW can be referred from latest version of HSE Firmware reference manual.