# NV Access Functions

Monday, May 18, 2015 8:50 PM

## NV Functions to be provided by MKOPA

### Function Prototypes

```
byte NV_AddKey( byte KeyType, byte Ksn, byte *pKey );

void NV_SetSecurityState( byte State );

byte NV_GetSecurityState();

byte NV_SetActiveKsn( byte KeyType, byte Ksn );

void NV_UpdateFrameCounter( byte KeyType,
                            byte Ksn,
                            byte FrameCounterType,
                            dword FrameCounterValue );

byte NV_GetActiveKsn( byte KeyType );

byte *NV_GetKey(byte KeyType, byte Ksn);
dword NV_GetFrameCounter(byte KeyType,
                         byte Ksn,
                         byte FrameCounterType);

byte *NV_GetSerialNumber();
```

### Definitions

#### Key Type

| | |
|---|---|
| #define KEY_TYPE_MASTER_KEY | 0x01 |
| #define KEY_TYPE_APP_DATA_KEY | 0x02 |

#### Frame Counter Type

| | |
|---|---|
| #define FRAME_COUNTER_INCOMING | 0x01 |
| #define FRAME_COUNTER_OUTGOING | 0x02 |

### Structure Definitions

#### Key Table Entry

```
typedef struct kte
{
 dword OutgoingFrameCounter;
 dword IncomingFrameCounter;
 byte Key[16];
 byte KeySequenceNumber;
} kte_t;
```

## byte NV_AddKey( byte KeyType, byte Ksn, byte *pKey );

### Parameters

KeyType - The Type of Key being added, KEY_TYPE_MASTER_KEY or KEY_TYPE_APP_DATA_KEY.
Ksn - The Key Sequence Number for this Key.
pKey - A pointer to the Key.

### Return Value

Status - 0x00 If Key was Successfully added, or 0x01 if it failed.

### When Called

This function will be called upon receipt of a Transport Key Request and is used to add a new key to the Key Table in NV RAM.

The Key Table stores two keys for each type of Key (Master or Data Key), one is the currently active key the other is unused. When a new key is added it will always be added to the unused entry in the Key Table. Also when a new Key is added the Incoming and Outgoing Frame Counters are set to 0.

## void NV_SetSecurityState( byte State );

### Parameters

State - The new security state to be set. E.g. Security Enabled = TRUE, Security Disabled = FALSE.

### When Called

This function is called upon receipt of a Set Security Request from the Trust Centre. It is used to Enable or Disable the security processing within the IBIS 2 device.

## byte NV_GetSecurityState();

### Return Value

Returns the current Security state of the device. TRUE = Security Enabled, FALSE = Security Disabled.

### When Called

This function is called by the message processing tasks in the IBIS 2 firmware to determine if Security Processing is enabled and whether commands need to be encrypted or decrypted before transmission.

## byte NV_SetActiveKsn( byte KeyType, byte Ksn );

### Parameters

KeyType - The Type of Key being made active, KEY_TYPE_MASTER_KEY or KEY_TYPE_APP_DATA_KEY.
Ksn - The Key Sequence Number for this Key.

### Return Value

Returns the Status indicating if the Active KSN was set.

### When Called

This function is called upon receipt of a Switch Key Request. It will first check to ensure that there is a Key with the given KSN in the appropriate Key Table. If no key is found in will return the "No Key" (0x02) Error Code. If a Key is found then it will set the Active (Master or Data) Key Sequence Number to the given KSN and will return the status "Success" (0x00).

## void NV_UpdateFrameCounter( byte KeyType, byte Ksn, byte FrameCounterType, dword FrameCounterValue );

### Parameters

KeyType - The Type of Key whose Frame Counter is being updated, KEY_TYPE_MASTER_KEY or KEY_TYPE_APP_DATA_KEY.
Ksn - The Key Sequence Number for this Key.
FrameCounterType - The Type of Frame Counter being updated. FRAME_COUNTER_INCOMING or FRAME_COUNTER_OUTGOING.
FrameCounterValue - The new value of the Frame Counter to be Stored.

### When Called

This function is called by the Encryption/Decryption routines to store a new frame counter value after processing a new frame.

Note, this function will be called every time a message is encrypted or decrypted. During busy periods of messaging this could result in too many writes to the NV RAM which could cause page wear problems. This function should protect against this sort of problem by implementing some kind of delayed write mechanism. See the section "Recommendations for storage of Frame Counters" in the IBIS 2 Messaging Protocol document for further details.

## byte NV_GetActiveKsn( byte KeyType )

### Parameters

KeyType - The Type of Key to get, KEY_TYPE_MASTER_KEY or KEY_TYPE_APP_DATA_KEY.

### Return Value

The Active KSN for the given Key Type

### When Called

This function is called by the Encryption/Decryption routines to get the Active Key Sequence Number from the IB for the given Key Type.

## byte *NV_GetKey( byte KeyType, byte Ksn )

### Parameters

KeyType - The Type of Key to get, KEY_TYPE_MASTER_KEY or KEY_TYPE_APP_DATA_KEY.
Ksn - The Key Sequence Number of Key to get.

### Return Value

A pointer to a Key.

### When Called

This function is called by the Encryption/Decryption routines to get the Keying information required to process a secured message. The function will use the given Key Sequence Number and search within the appropriate Key Table for a Key with that KSN. If a Key is found it will return a pointer to the Key Table entry. If no Key is found it will return NULL.

## byte *NV_GetSerialNumber( )

### Return Value
A pointer to a byte array containing the serial number of the device.

### When Called
This function is used by the routing and security software to get the serial number of the device.

## dword NV_GetFrameCounter( byte KeyType, byte Ksn, byte FrameCounterType )

### Parameters
KeyType - The Type of Key to get, KEY_TYPE_MASTER_KEY or KEY_TYPE_APP_DATA_KEY.
Ksn - The Key Sequence Number of Frame Counter to get.
FrameCounterType - The Type of Frame Counter being updated. FRAME_COUNTER_INCOMING or FRAME_COUNTER_OUTGOING.

### Return Value
The latest Frame Counter value from the RAM based copy.

### When Called
This function is called by the Encryption/Decryption routines to get the Frame Counter required to process a secured message. The function will use the given Key Sequence Number and search within the RAM based copy of the Frame Counters for the value that is associated with the KSN and frame counter type. If a frame counter is found it will return the frame counter value. If no frame counter is found it will return 0xFFFFFFFF.