Microsoft

Azure RTOS NetX Crypto User Guide

Published: February 2020

For the latest information, please see
azure.com/rtos

Part Number: 000-1059
Revision 6.0

Contents

Chapter 1

Introduction to NetX Crypto

NetX Crypto is a high-performance real-time implementation of cryptographic algorithms designed to provide data encryption and authentication services. NetX Crypto is designed to plug in for NetX Secure TLS, DTLS, and IPsec modules. Applications may also use NetX Crypto as a standalone module outside network security.

NetX Crypto Unique Features

NetX Crypto is implemented in the standard C language (C99), compatible with virtually all C/C++ compilers. Its modular design allows an application to only link in the crypto algorithms it needs to use, therefore achieving minimal code size. The implementation is designed to work with most 32-bit microprocessors and uses only the basic math operations (addition, subtraction, multiplication, division, logical AND, OR, NOR, and bit shift operations). All these operations are used with 32-bit quantities, making NetX Crypto portable across most 32-bit microprocessors. The implementation is specifically optimized to run on resource constrained microprocessors, targeting deeply embedded applications.

Algorithms supported by NetX Crypto

NetX Crypto supports the following cryptographic algorithms. NetX Crypto follows all general recommendations and basic requirements within the constraints of a real-time operating system and platforms requiring a small memory footprint and efficient execution.

Algorithm	Key Length (bits)
AES(CBC, CTR)	128, 192, 256
AES(XCBC)	128
AES-CCM 8	128
3DES(CBC)	192
HMAC-SHA1	Any length
HMAC-SHA224	Any length
HMAC-SHA256	Any length
HMAC-SHA384	Any length
HMAC-SHA512	Any length
HMAC-SHA512/224	Any length
HMAC-SHA512/256	Any length
HMAC-MD5	Any length
RSA	1024, 2048, 3072, 4096

Algorithm	Digest Length (bits)	Block Size (bits)
SHA1	160	512
SHA224	224	512
SHA256	256	512
SHA384	384	1024
SHA512	512	1024
MD5	128	512
HMAC-SHA1	160	512
HMAC-SHA224	224	512
HMAC-SHA256	256	512
HMAC-SHA384	384	1024
HMAC-SHA512	512	1024
HMAC-SHA512/224	224	1024
HMAC-SHA512/256	256	1024
HMAC-MD5	128	512
Elliptic Curve	P192/224/256/384/521	

NetX Crypto Constraints

None.

Chapter 2

Installation and Use of NetX Crypto

This chapter contains a description of various issues related to installation, setup, and usage of the NetX Crypto component.

Product Distribution

NetX Crypto is shipped on a single CD-ROM compatible disk. The package includes source files, include files, and a PDF file that contains this document, as follows:

nx_crypto.h	Public API header file NetX Crypto module
nx_crypto_*.c/h	C/H Source files for NetX Crypto
NetX_Crypto_User_Guide.pdf	PDF description of NetX Crypto Module.

NetX Crypto Installation

In order to use NetX Crypto, the entire distribution mentioned previously should be copied to the same directory level where NetX is installed. For example, if NetX is installed in the directory "*threadx**arm7**NetX*" then the *nx_crypto*.** directories should be copied into "*threadx**arm7**NetXCrypto*".

Build NetX Crypto Library

NetX Crypto library has been validated with ARM Cortex M4 CPU and IAR version 8 tool chain.

Configuration Options

There are several configuration options for building NetX Crypto. Following is a list of all options, where each is described in detail:

Define	Meaning
NX_CRYPTO_MAX_RSA_MODULUS_SIZE	Defined, this option gives the maximum RSA modulus expected, in bits. The default value is 4096 for a 4096-bit modulus. Other values can be 3072, 2048, or 1024 (not recommended).
NX_CRYPTO_FIPS	Defined, this option enables extra security features required for FIPS-Compliant usage. This option is not enabled for non-FIPS build.

Chapter 3

Functional Description of NetX Crypto

Execution Overview

This chapter contains a functional description of NetX Crypto. There are two primary types of program execution in a NetX Crypto application: initialization and application interface calls.

NetX Crypto can be used as a standalone cryptographic library, or can be used with ThreadX, NetX, and/or NetX Secure.

AES

Algorithm Standard: NetX Crypto implements AES according to NIST FIPS 197, which can be found at:
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Key Lengths Supported: 128, 192, 256

Modes Supported: CBC, CTR, (Key length 128-, 192-, 256-bit)
XCBC (key length 128-bit only),
CCM8 (key length 128-bit only)

Memory Requirements: Application specifies input buffer and output buffer and an AES control structure. The AES control structure maintains AES algorithm state between calls to the API. The input buffer contains data to be encrypted or decrypted, and can be arbitrary size. The output buffer is used by AES to store data being processed by AES. The output buffer size must be no smaller than the input buffer size, and must be a multiple of 16 bytes, the AES block size. The input and output buffers must be contiguous memory and may not overlap, except in the special case of encrypting in-place (using the same memory for input and output). When encrypting in-place, the output buffer starts at exactly the same location as the input buffer, and must be no smaller than the input buffer. When AES encryption operates in-place no extra scratch memory is required.

3DES

Algorithm Standard: NetX Crypto implements Tripple DES(TDES, also known as 3DES) according to NIST Special Publication 800-67 rev 2: *"Recommendataion for the Triple Data Encryption Algorithm (TDES) Block Cipher"*, which can be found at:

https://csrc.nist.gov/publications/detail/sp/800-67/rev-2/final

Key Length Supported: 64 * 3 = 192

Memory Requreiment: None

In NetX Crypto, the term "3DES" is used interchangeably with "TDES".

MD5

Algorithm Standard: NetX Crypto implements MD5 according to RFC 1321: *"The MD5 Message-Digest Algorithm"*

Memory Requirement: The application must supply an MD5 control block structure, used to maintain state between MD5 operations.

SHA1, SHA256/512

Algorithm Standard: NetX Crypto implements SHA1/256/512 according to NIST FIPS publication 180-4: "*Secure Hash Standard*", which can be found at:

http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

Hash block size:
SHA1: 160 bits hash value
SHA 224: 224 bits hash value
SHA 256: 256 bits hash value
SHA 384: 384 bits hash value
SHA 512: 512 bits hash value
SHA 512/224: 224 bits hash value
SHA 512/256: 256 bits hash value

In NetX Crypto, SHA256 routines are used to hadn SHA256 and SHA224. SHA512 routines are used to hand SHA512, SHA384, SHA512/224 and SHA512/256.

Memory Requirement: The application must provide a SHA control block structure for maintaining state between operations.

RSA

Standard: NetX Crypto implements RSA according to the standard "*PKCS #1 v2.2: RSA Cryptography Standard*", which is published as RFC 8017 and can also be found at:

https://www.emc.com/collateral/white-papers/h11300-pkcs-1v2-2-rsa-cryptography-standard-wp.pdf

Memory Requirement: The application must provide an RSA control block structure for maintaining state between operations and to provide necessary "scratch" buffer space for intermediate calculations.

HMAC

Standard: NetX Crypto implements HMAC according to FIPS PUB 198-1: "*The Keyed-Hash Message Authentication Code (HMAC)*", which can be found at:

https://csrc.nist.gov/csrc/media/publications/fips/198/1/final/documents/fips-198-1_final.pdf

Memory Requirement: The application must provide an HMAC control block structure for maintaining state between operations. The actual control block supplied depends on the desired underlying hash operation (e.g. SHA1, MD5).

Elliptic Curve

Standard: NetX Crypto implements Elliptic Curve. The supported named curves are (prime field only):
P-192
P-224
P-256
P-384
P-521

Uncompressed format is supportd. See section 2.3.3 and 2.3.4 of SEC1-v1:

http://www.secg.org/sec1-v2.pdf

Memory Requirement: None

ECDSA

Standard: NetX Crypto implements ECDSA according to FIPS PUB 186-4: "*Digital Signature Standard (DSS)*", which can be found at:

https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf

Memory Requirement: The application must provide an ECDSA control block structure for maintaining state between operations.

ECDH

Standard: NetX Crypto implements ECDH according to FIPS PUB 800-56Ar2: "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", which can be found at:

https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-56ar2.pdf

Memory Requirement: The application must provide an ECDH control block structure for maintaining state between operations.

DRBG

Standard: NetX Crypto implements DRBG according to FIPS PUB 800-90Ar1: "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", which can be found at:

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

Memory Requirement: The application must provide an DRBG control block structure for maintaining state between operations.

FIPS-Compliant

NetX Crypto FIPS 140-2

Chapter 4

NetX Crypto API Description

nx_crypto_initialize

Initializes the NetX Secure Library

Prototype

```
UINT  nx_crypto_initialize(VOID)
```

Description

This function initializes the NetX Crypto library module. Before using any of the other cryptographic functions, the application must call this function to perform initializion and tovalidate the integrity of the library. Failure to call this function before using other NetX Crypto services will result in errors being returned.

Parameters

None

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialized NetX Crypto library. The library functions are now ready, and can be used.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library fails to initialize, or fails the integrity check. Application cannot use this library.

Example

TODO

nx_crypto_module_state_get

Retrieve the current status of the FIPS-enabled module

Prototype

```
UINT  nx_crypto_module_state_get(VOID)
```

Description

This service is only available in the FIPS build library. It returns the state of the current state of the NetX Crypto library.

Parameters

None

Return Values

Status Flag:

NX_CRYPTO_LIBRARY_STATE_UNINITIALIZED	0x00000001
NX_CRYPTO_LIBRARY_STATE_POST_IN_PROGRESS	0x00000002
NX_CRYPTO_LIBRARY_STATE_INTEGRITY_TEST_FAILED	0x00000004
NX_CRYPTO_LIBRARY_STATE_FUNCTIONAL_TEST_FAILED	0x00000008
NX_CRYPTO_LIBRARY_STATE_OPERATIONAL	0x80000000

All other values are invalid.

Example

TODO

_nx_crypto_method_aes_init

Initializes the AES crypto control block

Prototype

```
UINT _nx_crypto_method_aes_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the AES control block with the given key string. Once the AES control block is initialized, subsequent AES operation will be using the same key and key size.

Application may create multiple AES control blocks, each represents a session. A key is assigned to a control block. Subsequent encryption or decryption operation can reference to the same AES control block without the need to re-initialize the AES control block. If the key for the session is changed, application needs to re-initialize AES control block with the updated key.

Calling *_nx_crypto_method_aes_init()* automatically updates a previously configured key and key size to the new key.

Parameters

method	Pointer to a valid AES crypto method control block. The following pre-defined AES crypto methods are available: *crypto_method_aes_cbc_128* *crypto_method_aes_cbc_192* *crypto_method_aes_cbc_256* *crypto_method_aes_ctr_128* *crypto_method_aes_ctr_192* *crypto_method_aes_ctr_256* *crypto_method_aes_xcbc_128* *crypto_method_aes_ccm_8_128*
key	Points to a buffer containing the AES key
key_size_in_bits	Size of the key, in bits. Valid values are: *NX_CRYPTO_AES_KEY_SIZE_128_BITS* *NX_CRYPTO_AES_KEY_SIZE_192_BITS* *NX_CRYPTO_AES_KEY_SIZE_256_BITS*

All other values are invalid.

handle This service returns a handle to the caller. The handle is implementation-dependent, and is not being used in this imiplementation. Application shall pass NULL for the handle.

crypto_metadata Pointer to a valid memory space for the AES control block. The starting address of the memory space must be 4-byte aligned.

crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For AES, the metadata size must be *sizeof(NX_AES)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the AES control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.
NX_CRYPTO_UNSUPPORTED_KEY_SIZE	(0x20002)	Key size is not a valid for AES.

Example

_nx_crypto_method_aes_operation

Perform an AES operation (encryption or decryption).

Prototype

```
UINT _nx_crypto_method_aes_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs AES encryption or decryption operation. The AES control block must have been initialized with *_nx_crypto_method_aes_init()*. The AES algorithm to be performed is based on the algorithm specified in the *method* control block.

The input buffer size must be a multiple of 16 bytes. The size of the decrypted data is the same size of the input data size. If the encrypted data was padded to achieve an even multiple of the AES block size, the padding will be included in the output buffer and must be handled by the application.

This operation does not keep state information, and does not alter the key material in the AES control block.

When the op is NX_CRYPTO_SET_ADDITIONAL_DATA and algoritm is AES-CCM8, the input points to additional data and input_length_in_byte is the length of additional data.

Parameters

op	Type of operation to perform. Valid opertions are: *NX_CRYPTO_ENCRYPT* *NX_CRYPTO_DECRYPT* *NX_CRYPTO_AUTHENTICATE (AES-XCBC only)* *NX_CRYPTO_SET_ADDITIONAL_DATA (AES-CCM8 only)*

handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid AES crypto method. The crypto method used here must be the same used in the *nx_crypto_method_aes_init().*
input_data	Points to a buffer containing encrypted text data. There are not restrictions on input buffer.
input_data_size	Size of the input data, in bytes. The input data size must be a multiple of 16 bytes.
iv_ptr	Pointer to the Initial Vector. There are no restrictions on the IV buffer.
iv_size	Size of the Initial Vector block, in bytes This field must be 16.
output_buffer	Pointer to the memory area for AES to store the clear text data. Application must allocate space for the output buffer. Output buffer may overlap with input buffer. The output buffer may overlap with the input buffer if they share the same starting address.
output_buffer_size	Size of the output buffer in bytes. Output buffer size must be at least the same of the input buffer size, and the output buffer size must be a multiple of 16 bytes.
crypto_metadata	Pointer to the AES control block used in *_nx_crypto_method_aes_init()*.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For AES, the metadata size must *sizeof(NX_AES)*
packet_ptr	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
nx_crypto_hw_process_callback	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the AES operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid AES algorithm being specified.

Example

_nx_crypto_method_aes_cleanup

Clean up the AES control block.

Prototype

```
UINT  _nx_crypto_method_aes_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the AES control block after it determins this AES session is no longer needed.

Parameters

crypto_metadata Pointer to the AES control block used in *_nx_crypto_method_aes_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the AES session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_3des_init

Initialize the 3DES control block.

Prototype

```
UINT _nx_crypto_method_3des_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the Triple DES (3DES) control block with the given three key strings. The key strings must be 8 bytes each. The three DES keys must be concatenated into contiguous memory of 24-byte buffer. For FIPS-compliant build, the three keys must be different from each or the function will return the NX_CRYPTO_INVALID_KEY error. Once the 3DES control block is initialized, subsequent 3DES operations will use the same keys.

An application may create multiple 3DES control blocks, each representing a session. A key is assigned to a control block and subsequent encryption or decryption operations can reference the same control block without needing to re-initialize. If the key for a session is changed, the application will need to re-initialize the control block with the updated key.

Calling *_nx_crypto_method_3des_init()* automatically updates a previously configured key to the new keys.

Parameters

method	Pointer to a valid 3DES crypto method control block. The following pre-defined 3DES crypto method is available: *crypto_method_3des*
key	Points to a buffer containing the three (3) DES key
key_size_in_bits	Must be 192 (3 keys, each 64 bits).
handle	This service returns a handle to the caller. The handle identifies the 3DES control block being initialized. Subsequent 3DES operations (encryption, decryption, and cleanup) use this handle to access the 3DES control block

crypto_metadata Pointer to a valid memory space for the 3DES control block. The starting address of the memory space must be 4-byte aligned.

crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For 3DES, the metadata size must be *sizeof(NX_3DES)*

Return Value

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the 3DES control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.
NX_CRYPTO_UNSUPPORTED_KEY_SIZE	(0x20002)	Key size is not a valid for 3DES.

Example

_nx_crypto_method_3des_operation

Encrypt or Decrypto with 3DES.

Prototype

```
UINT  _nx_crypto_method_3des_operation(UINT op,
                       VOID *handle,
                       struct NX_CRYPTO_METHOD_STRUCT *method,
                       UCHAR *key,
                       NX_CRYPTO_KEY_SIZE key_size_in_bits,
                       UCHAR *input,
                       ULONG input_length_in_byte,
                       UCHAR *iv_ptr,
                       UCHAR *output,
                       ULONG output_length_in_byte,
                       VOID *crypto_metadata,
                       ULONG crypto_metadata_size,
                       VOID *packet_ptr,
                       VOID (*nx_crypto_hw_process_callback)(VOID
                             *packet_ptr, UINT status))
```

Description

This function performs 3DES encryption or decryption operation. The 3DES control block must have been initialized with *_nx_crypto_method_3des_init()*. The 3DES algorithm to be performed is based on the algorithm specified in the *method* control block.

The input buffer size must be a multiple of 8 bytes. The size of the decrypted data is the same size of the input data size. If the encrypted data was padded to achieve an even multiple of the 3DES block size, the padding will be included in the output buffer and must be handled by the application.

This operation does not keep state information, and does not alter the key material in the 3DES control block.

Parameters

op	Type of operation to perform. Valid opertions are: *NX_CRYPTO_ENCRYPT* *NX_CRYPTO_DECRYPT*
handle	The handle initialized by *_nx_crypto_method_3des_init()*.
method	Pointer to the valid 3DES crypto method. The crypto method used here must be the same used in the *nx_crypto_method_3des_init().*
input_data	Points to a buffer containing encrypted text data. There are not restrictions on input buffer.

input_data_size Size of the input data, in bytes. The input data size must be a multiple of 8 bytes.

iv_ptr Pointer to the Initial Vector. There are no restrictions on the IV buffer.

iv_size Size of the Initial Vector block, in bytes This field must be 8.

output_buffer Pointer to the memory area for 3DES to store the clear text data. Application must allocate space for the output buffer. Output buffer may overlap with input buffer. The output buffer may overlap with the input buffer if they share the same starting address.

output_buffer_size
Size of the output buffer in bytes. Output buffer size must be at least the same of the input buffer size, and the output buffer size must be a multiple of 8 bytes.

crypto_metadata Pointer to the 3DES control block used in *_nx_crypto_method_3des_init()*.

crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For 3DES, the metadata size must be *sizeof(NX_3DES)*

packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

nx_crypto_hw_process_callback
This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Description

This function performs 3DES encryption. The 3DES control block must have been initialized with *_nx_crypto_moethod_3des_init()*. This operation does not keep state information, and does not alter the key material in the 3DES control block. Note that padding is not added by this function so the caller will need to handle padding before invoking the encryption operation.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the 3DES control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_3des_cleanup

Clean up the 3DES control block.

Prototype

```
UINT  _nx_crypto_method_3des_cleanup(VOID *crypto_metadata)
```

Description

Application calls this function to clean up the 3DES control block after it determins this 3DES session is no longer needed.

Parameters

handle	The handle initialized by *_nx_crypto_method_3des_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the 3DES session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_drbg_init

Initializes the DRBG crypto control block

Prototype

```
UINT _nx_crypto_method_drbg_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the DRBG control block with the given key string. Once the DRBG control block is initialized, subsequent DRBG operation shall be using the same control block.

Application may create multiple DRBG control blocks, each represents a session. Initializing the DRBG control block starts a new hash computation session. Re-initializing the DRBG control block abandons the current session and stars a new one.

Parameters

method	Pointer to a valid DRBG crypto method control block. The following pre-defined crypto methods are available: *crypto_method_drbg*
key	This field is not used for DRBG.
key_size_in_bits	This field is not used for DRBG.
handle	This service returns a handle to the caller. The handle is implementation-dependent and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the DRBG control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For DRBG, the metadata size must be *sizeof(NX_CRYPTO_DRBG)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the DRBG control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_drbg_operation

Perform DRBG operation

Prototype

```
UINT  __nx_crypto_method_drbg_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs DRBG operation. The DRBG control block must have been initialized with *_nx_crypto_method_drbg_init()*. The DRBG algorithm to be performed is based on the algorithm specified in the *method* control block. By default AES-128 is used for DRBG.

When the operation is NX_CRYPTO_DRBG_OPTIONS_SET, the input points to NX_CRYPTO_DRBG_OPTIONS structure. When the operation is NX_CRYPTO_DRBG_INSTANTIATE, the key points to nonce, input points to personalization string. When the operation is NX_CRYPTO_DRBG_RESEED or NX_CRYPTO_DRBG_GENERATE, the input points to additional input.

Parameters

op	Type of operation to perform. Valid operation is: *NX_CRYPTO_DRBG_OPTIONS_SET* *NX_CRYPTO_DRBG_INSTANTIATE* *NX_CRYPTO_DRBG_RESEED* *NX_CRYPTO_DRBG_GENERATE*
handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid DRBG crypto method. The crypto method used here must be the same used in the *_nx_crypto_method_drbg_init().*
key	Pointer to the the nonce for the instantiate operation. This field is not used for other operations.

key_size_in_bits Size of the nonce, in bits. This field is not used for other operations.

input When op is NX_CRYPTO_DRBG_OPTIONS_SET, this field points to DRBG options. When op is NX_CRYPTO_DRBG_INSTANTIATE, this field points to personalization string. When op is NX_CRYPTO_DRBG_RESEED or NX_CRYPTO_DRBG_GENERATE, this field points to additional input data.

input_length_in_byte Size of the input data, in bytes.

iv_ptr This field is not used for DRBG.

output When op is NX_CRYPTO_DRBG_GENERATE, this field points to the memory area for the generated DRBG. Otherwise, this field is not used.

output_length_in_byte Size of the output buffer in bytes.

crypto_metadata Pointer to the DRBG control block used in *_nx_crypto_method_drbg_init()*.

crypto_metadata_size Size, in bytes, of the crypto_metadata area. For DRBG, the metadata size must *sizeof(NX_CRYPTO_DRBG)*

packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

nx_crypto_hw_process_callback This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the DRBG operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid DRBG algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_drbg_cleanup

Clean up the DRBG control block.

Prototype

```
UINT  _nx_crypto_method_drbg_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the DRBG control block after it determines this DRBG session is no longer needed.

Parameters

crypto_metadata Pointer to the DRBG control block used in *_nx_crypto_method_drbg_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the DRBG session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_ecdh_init

Initializes the ECDH crypto control block

Prototype

```
UINT _nx_crypto_method_ecdh_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the ECDH control block with the given key string. Once the ECDH control block is initialized, subsequent ECDH operation shall be using the same control block.

Application may create multiple ECDH control blocks, each represents a session. Initializing the ECDH control block starts a new hash computation session. Re-initializing the ECDH control block abandons the current session and stars a new one.

Parameters

method	Pointer to a valid ECDH crypto method control block. The following pre-defined crypto methods are available: *crypto_method_ecdh*
key	This field is not used for ECDH.
key_size_in_bits	This field is not used for ECDH.
handle	This service returns a handle to the caller. The handle is implementation-dependent and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the ECDH control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For ECDH, the metadata size must be *sizeof(NX_CRYPTO_ECDH)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the ECDH control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_ecdh_operation

Perform ECDH operation

Prototype

```
UINT _nx_crypto_method_ecdh_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs ECDH operation. The ECDH control block must have been initialized with *_nx_crypto_method_ecdh_init()*. The ECDH algorithm to be performed is based on the algorithm specified in the *method* control block.

When the operation is NX_CRYPTO_EC_CURVE_SET, the input points to Elliptic Curve crypto method. When the operation is NX_CRYPTO_EC_KEY_PAIR_GENERATE, the output points to NX_CRYPTO_EXTENDED_OUTPUT structure and the key pair is copied to nx_crypto_extended_output_data. When the operation is NX_CRYPTO_DH_SETUP, the public key is returned to nx_crypto_extended_output_data. When the operation is NX_CRYPTO_DH_KEY_PAIR_IMPORT, the input points to public key and key points to private key. When the operation is NX_CRYPTO_DH_PRIVATE_KEY_EXPORT, the private key is copied to nx_crypto_extended_output_data. When the operation is NX_CRYPTO_DH_CALCULATE, the input points to remote public key and the shared secret is copied to nx_crypto_extended_output_data.

Parameters

op Type of operation to perform. Valid operation is:

NX_CRYPTO_EC_CURVE_SET
NX_CRYPTO_DH_SETUP
NX_CRYPTO_DH_CALCULATE
NX_CRYPTO_DH_KEY_PAIR_IMPOPRT
NX_CRYPTO_DH_KEY_PAIR_GENERATE

NX_CRYPTO_DH_PRIVATE_KEY_EXPORT

handle This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

method Pointer to the valid ECDH crypto method. The crypto method used here must be the same used in the *_nx_crypto_method_ecdh_init().*

key When op is NX_CRYPTO_DH_IMPORT_KEY_PAIR, this field points to private key. Otherwise, this field is not used for ECDH.

key_size_in_bits Size of the key, in bits.

input When op is NX_CRYPTO_EC_CURVE_SET, this field points to Elliptic Curve crypto method. When op is NX_CRYPTO_DH_SETUP, this field is not used. When op is NX_CRYPTO_DH_CALCULATE, this field points to a buffer containing input text data. When op is NX_CRYPTO_DH_IMPORT_KEY_PAIR, this field points to public key.

input_length_in_byte Size of the input data, in bytes.

iv_ptr This field is not used for ECDH.

output When op is NX_CRYPTO_EC_CURVE_SET or NX_CRYPTO_DH_IMPORT_KEY_PAIR, this field is not used. When op is NX_CRYPTO_DH_SETUP, this field points to the memory area for the generated ECDH public key. When op is NX_CRYPTO_DH_CALCULATE, this field points to the memory area for the generated ECDH shared secret.

output_length_in_byte Size of the output buffer in bytes.

crypto_metadata Pointer to the ECDH control block used in *_nx_crypto_method_ecdh_init()*.

crypto_metadata_size Size, in bytes, of the crypto_metadata area. For ECDH, the metadata size must *sizeof(NX_CRYPTO_ECDH)*

packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

nx_crypto_hw_process_callback This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the ECDH operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid ECDH algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_ecdh_cleanup

Clean up the ECDH control block.

Prototype

```
UINT _nx_crypto_method_ecdh_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the ECDH control block after it determines this ECDH session is no longer needed.

Parameters

crypto_metadata Pointer to the ECDH control block used in *_nx_crypto_method_ecdh_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the ECDH session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_ecdsa_init

Initializes the ECDSA crypto control block

Prototype

```
UINT _nx_crypto_method_ecdsa_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the ECDSA control block with the given key string. Once the ECDSA control block is initialized, subsequent ECDSA operation shall be using the same control block.

Application may create multiple ECDSA control blocks, each represents a session. Initializing the ECDSA control block starts a new hash computation session. Re-initializing the ECDSA control block abandons the current session and stars a new one.

Parameters

method	Pointer to a valid ECDSA crypto method control block. The following pre-defined crypto methods are available: *crypto_method_ecdsa*
key	This field is not used for ECDSA.
key_size_in_bits	This field is not used for ECDSA.
handle	This service returns a handle to the caller. The handle is implementation-dependent and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the ECDSA control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For ECDSA, the metadata size must be *sizeof(NX_CRYPTO_ECDSA)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the ECDSA control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_ecdsa_operation

Perform ECDSA operation

Prototype

```
UINT _nx_crypto_method_ecdsa_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs ECDSA operation. The ECDSA control block must have been initialized with *_nx_crypto_method_ecdsa_init()*. The ECDSA algorithm to be performed is based on the algorithm specified in the *method* control block.

When the operation is NX_CRYPTO_EC_CURVE_SET, the input points to Elliptic Curve crypto method. When the operation is NX_CRYPTO_EC_KEY_PAIR_GENERATE, the output points to NX_CRYPTO_EXTENDED_OUTPUT structure and the key pair is copied to nx_crypto_extended_output_data.

Parameters

op	Type of operation to perform. Valid operation is: *NX_CRYPTO_EC_CURVE_SET* *NX_CRYPTO_AUTHENTICATE* *NX_CRYPTO_VERIFY*
handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid ECDSA crypto method. The crypto method used here must be the same used in the *_nx_crypto_method_ecdsa_init().*
key	Points to the key when op is NX_CRYPTO_VERIFY. There are not restrictions on key buffer. This field is not used for other values of op.
key_size_in_bits	Size of the key, in bits.

input When op is NX_CRYPTO_EC_CURVE_SET, this field points to Elliptic Curve crypto method. Otherwise, this field points to a buffer containing input text data.

input_length_in_byte
Size of the input data, in bytes.

iv_ptr This field is not used for ECDSA.

output When op is NX_CRYPTO_EC_CURVE_SET, this field is not used. When op is NX_CRYPTO_AUTHENTICATE, this field points to the memory area for the generated ECDSA signature. When op is NX_CRYPTO_VERIFY, this field points to the memory area for the verified ECDSA signature.

output_length_in_byte
Size of the output buffer in bytes.

crypto_metadata Pointer to the ECDSA control block used in *_nx_crypto_method_ecdsa_init()*.

crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For ECDSA, the metadata size must *sizeof(NX_CRYPTO_ECDSA)*

packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

nx_crypto_hw_process_callback
This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the ECDSA operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid ECDSA algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_ecdsa_cleanup

Clean up the ECDSA control block.

Prototype

```
UINT _nx_crypto_method_ecdsa_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the ECDSA control block after it determines this ECDSA session is no longer needed.

Parameters

crypto_metadata Pointer to the ECDSA control block used in *_nx_crypto_method_ecdsa_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the ECDSA session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_hmac_md5_init

Initializes the HMAC MD5 crypto control block

Prototype

```
UINT _nx_crypto_method_hmac_md5_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the HMAC MD5 control block with the given key string. Once the HMAC MD5 control block is initialized, subsequent HMAC MD5 operation shall be using the same control block.

Application may create multiple HMAC MD5 control blocks, each represents a session. Initializing the HMAC MD5 control block starts a new hash computation session. Re-initializing the HMAC MD5 control block abandons the current session and stars a new one.

Parameters

method	Pointer to a valid HMAC MD5 crypto method control block. The following pre-defined crypto methods are available: *crypto_method_hmac_md5*
key	Points to the key. There are not restrictions on key buffer.
key_size_in_bits	Size of the key, in bits.
handle	This service returns a handle to the caller. The handle is implementation-dependent and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the HMAC MD5 control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For HMAC MD5, the metadata size must be *sizeof(NX_CRYPTO_MD5_HMAC)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the HMAC MD5 control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_hmac_md5_operation

Perform an HMAC MD5 hash operation.

Prototype

```
UINT _nx_crypto_method_hmac_md5_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs HMAC MD5 hash operation. The HMAC MD5 control block must have been initialized with *_nx_crypto_method_hmac_md5_init()*. The HMAC MD5 algorithm to be performed is based on the algorithm specified in the *method* control block.

For the final *NX_CRYPTO_HASH_CALCULATE* operation, the output buffer size must be 16 bytes.

This operation does not keep state information, and does not alter the key material in the HMAC MD5 control block.

Parameters

op	Type of operation to perform. Valid operation is: *NX_CRYPTO_HASH_INITIALIZE* *NX_CRYPTO_HASH_UPDATE* *NX_CRYPTO_HASH_CALCULATE*
handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid HMAC MD5 crypto method. The crypto method used here must be the same used in the *nx_crypto_method_hmac_md5_init().*
key	Points to the key. There are not restrictions on key buffer.
key_size_in_bits	Size of the key, in bits.

input_data	Points to a buffer containing input text data. There are not restrictions on input buffer.
input_data_size	Size of the input data, in bytes.
iv_ptr	This field is not used for HMAC MD5.
iv_size	This field is not used for HMAC MD5.
output_buffer	Pointer to the memory area for the generated HMAC MD5 hash.
output_buffer_size	Size of the output buffer in bytes.
crypto_metadata	Pointer to the HMAC MD5 control block used in *_nx_crypto_method_hmac_md5_init()*.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For HMAC MD5, the metadata size must *sizeof(NX_CRYPTO_MD5_HMAC)*
packet_ptr	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
nx_crypto_hw_process_callback	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the HMAC MD5 operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid HMAC MD5 algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_hmac_sha1_init

Initializes the HMAC SHA1 crypto control block

Prototype

```
UINT _nx_crypto_method_hmac_sha1_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the HMAC SHA1 control block with the given key string. Once the HMAC SHA1 control block is initialized, subsequent HMAC SHA1 operation shall be using the same control block.

Application may create multiple HMAC SHA1 control blocks, each represents a session. Initializing the HMAC SHA1 control block starts a new hash computation session. Re-initializing the HMAC SHA1 control block abandons the current session and stars a new one.
.

Parameters

method	Pointer to a valid HMAC SHA1 crypto method control block. The following pre-defined crypto methods are available: *crypto_method_hmac_sha1*
key	Points to the key. There are not restrictions on key buffer.
key_size_in_bits	Size of the key, in bits.
handle	This service returns a handle to the caller. The handle is implementation-dependent and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the HMAC SHA1 control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For HMAC SHA1, the metadata size must be *sizeof(NX_CRYPTO_SHA1_HMAC)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the HMAC SHA1control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_hmac_sha1_operation

Perform HMAC SHA1 hash operation

Prototype

```
UINT _nx_crypto_method_hmac_sha1_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs HMAC SHA1 hash operation. The HMAC SHA1 control block must have been initialized with *_nx_crypto_method_hmac_sha1_init()*. The HMAC SHA1 algorithm to be performed is based on the algorithm specified in the *method* control block.

For the final *NX_CRYPTO_HASH_CALCULATE* operation, the output buffer size must be 20 bytes.
.

Parameters

op	Type of operation to perform. Valid operation is: *NX_CRYPTO_HASH_INITIALIZE* *NX_CRYPTO_HASH_UPDATE* *NX_CRYPTO_HASH_CALCULATE*
handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid HMAC SHA1 crypto method. The crypto method used here must be the same used in the *_nx_crypto_method_hmac_sha1_init().*
key	Points to the key. There are not restrictions on key buffer.
key_size_in_bits	Size of the key, in bits.
input_data	Points to a buffer containing input text data. There are not restrictions on input buffer.
input_data_size	Size of the input data, in bytes.

iv_ptr This field is not used for HMAC SHA1.

iv_size This field is not used for HMAC SHA1.

output_buffer Pointer to the memory area for the generated HMAC SHA1 hash.

output_buffer_size
Size of the output buffer in bytes.

crypto_metadata Pointer to the HMAC SHA1 control block used in *_nx_crypto_method_hmac_sha1_init()*.

crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For HMAC SHA1, the metadata size must *sizeof(NX_CRYPTO_SHA1_HMAC)*

packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

nx_crypto_hw_process_callback
This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the HMAC SHA1 operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid HMAC SHA1 algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_hmac_sha1_cleanup

Clean up the HMAC SHA1 control block.

Prototype

```
UINT _nx_crypto_method_hmac_sha1_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the HMAC SHA1 control block after it determines this HMAC SHA1 session is no longer needed.

Parameters

crypto_metadata Pointer to the HMAC SHA1 control block used in *_nx_crypto_method_hmac_sha1_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the HMAC SHA1 session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_hmac_sha256_init

Initializes the HMAC SHA256 crypto control block

Prototype

```
UINT _nx_crypto_method_hmac_sha256_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the HMAC SHA256 control block with the given key string. Once the HMAC SHA256 control block is initialized, subsequent HMAC SHA256 operation shall be using the same control block.

Application may create multiple HMAC SHA256 control blocks, each represents a session. Initializing the HMAC SH256 control block starts a new hash computation session. Re-initializing the HMAC SHA256 control block abandons the current session and stars a new one with a new key.
.

Parameters

method	Pointer to a valid HMAC SHA256 crypto method control block. The following pre-defined crypto methods are available: *crypto_method_hmac_sha224* *crypto_method_hmac_sha256*
key	Points to the key. There are not restrictions on key buffer.
key_size_in_bits	Size of the key, in bits.
handle	This service returns a handle to the caller. The handle is implementation-dependent and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the HMAC SHA256 control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For HMAC SHA256, the metadata size must be *sizeof(NX_CRYTPO_SHA256_HMAC)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the HMAC SHA256 control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_hmac_sha256_operation

Perform HMAC SHA256 hash operation

Prototype

```
UINT _nx_crypto_method_hmac_sha256_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs HMAC SHA256 hash operation. The HMAC SHA256 control block must have been initialized with *_nx_crypto_method_hmac_sha256_init()*. The HMAC SHA256 algorithm to be performed is based on the algorithm specified in the *method* control block.

For the final *NX_CRYPTO_HASH_CALCULATE* operation, the output buffer size must be 32 bytes for SHA256, or 28 bytes for SHA224.
.

Parameters

op	Type of operation to perform. Valid operation is: *NX_CRYPTO_HASH_INITIALIZE* *NX_CRYPTO_HASH_UPDATE* *NX_CRYPTO_HASH_CALCULATE*
handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid HMAC SHA256 crypto method. The crypto method used here must be the same used in the *_nx_crypto_method_hmac_sha256_init().*
key	Points to the key. There are not restrictions on key buffer.
key_size_in_bits	Size of the key, in bits.
input_data	Points to a buffer containing input text data. There are not restrictions on input buffer.

input_data_size Size of the input data, in bytes.
iv_ptr This field is not used for HMAC SHA256.
iv_size This field is not used for HMAC SHA256.
output_buffer Pointer to the memory area for the generated HMAC SHA256 hash.
output_buffer_size
Size of the output buffer in bytes.
crypto_metadata Pointer to the HMAC SHA256 control block used in *_nx_crypto_method_hmac_sha256_init()*.
crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For HMAC SHA256, the metadata size must *sizeof(NX_CRYPTO_SHA256_HMAC)*
packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
nx_crypto_hw_process_callback
This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the HMAC SHA256 operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid HMAC SHA256 algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_hmac_sha256_cleanup

Clean up the HMAC SHA256 control block.

Prototype

```
UINT _nx_crypto_method_hmac_sha256_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the HMAC SHA256 control block after it determines this HMAC SHA256 session is no longer needed.

Parameters

crypto_metadata Pointer to the HMAC SHA256 control block used in *_nx_crypto_method_hmac_sha256_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the HMAC SHA256 session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_hmac_sha512_init

Initializes the HMAC SHA512 crypto control block

Prototype

```
UINT _nx_crypto_method_hmac_sha512_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the HMAC SHA512 control block with the given key string. Once the HMAC SHA512 control block is initialized, subsequent HMAC SHA512 operation shall be using the same control block.

Application may create multiple HMAC SHA512 control blocks, each represents a session. Initializing the HMAC SH512 control block starts a new hash computation session. Re-initializing the HMAC SHA512 control block abandons the current session and stars a new one with a new key.
.

Parameters

method	Pointer to a valid HMAC SHA512 crypto method control block. The following pre-defined crypto methods are available: *crypto_method_hmac_sha384* *crypto_method_hmac_sha512*
key	Points to the key. There are not restrictions on key buffer.
key_size_in_bits	Size of the key, in bits.
handle	This service returns a handle to the caller. The handle is implementation-dependent, and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the HMAC SHA512 control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For HMAC SHA512, the metadata size must be *sizeof(NX_CRYPTO_SHA512_HMAC)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the HMAC SHA512 control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_hmac_sha512_operation

Perform HMAC SHA512 hash operation

Prototype

```
UINT _nx_crypto_method_hmac_sha512_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs HMAC SHA512 hash operation. The HMAC SHA512 control block must have been initialized with *_nx_crypto_method_hmac_sha512_init()*. The HMAC SHA512 algorithm to be performed is based on the algorithm specified in the *method* control block.

For the final *NX_CRYPTO_HASH_CALCULATE* operation, the output buffer size must be 64 bytes for SHA512, or 48 bytes for SHA384.
.

Parameters

op	Type of operation to perform. Valid operation is: *NX_CRYPTO_HASH_INITIALIZE* *NX_CRYPTO_HASH_UPDATE* *NX_CRYPTO_HASH_CALCULATE*
handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid HMAC SHA512 crypto method. The crypto method used here must be the same used in the *_nx_crypto_method_hmac_sha512_init().*
key	Points to the key. There are not restrictions on key buffer.
key_size_in_bits	Size of the key, in bits.
input_data	Points to a buffer containing input text data. There are not restrictions on input buffer.

input_data_size Size of the input data, in bytes.
iv_ptr This field is not used for HMAC SHA512.

iv_size This field is not used for HMAC SHA512.
output_buffer Pointer to the memory area for the generated HMAC SHA512 hash.
output_buffer_size
Size of the output buffer in bytes.
crypto_metadata Pointer to the HMAC SHA512 control block used in *_nx_crypto_method_hmac_sha512_init()*.
crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For HMAC SHA512, the metadata size must *sizeof(NX_CRYPTO_SHA512_HMAC)*
packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
nx_crypto_hw_process_callback
This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the HMAC SHA512 operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid HMAC SHA512 algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_hmac_sha512_cleanup

Clean up the HMAC SHA512 control block.

Prototype

```
UINT _nx_crypto_method_hmac_sha512_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the HMAC SHA512 control block after it determines this HMAC SHA512 session is no longer needed.

Parameters

crypto_metadata Pointer to the HMAC SHA512 control block used in *_nx_crypto_method_hmac_sha512_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the HMAC SHA512 session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_md5_init

Initializes the MD5 crypto control block

Prototype

```
UINT _nx_crypto_method_md5_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the MD5 control block with the given key string. Once the MD5 control block is initialized, subsequent MD5 operation shall be using the same control block.

Application may create multiple MD5 control blocks, each represents a session. Initializing the MD5 control block starts a new hash computation session. Re-initializing the MD5 control block abandons the current session and stars a new one.

Parameters

method	Pointer to a valid MD5 crypto method control block. The following pre-defined crypto methods are available: *crypto_method_md5*
key	This field is not used for MD5.
key_size_in_bits	This field is not used for MD5
handle	This service returns a handle to the caller. The handle is implementation-dependent and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the MD5 control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For MD5, the metadata size must be *sizeof(NX_CRYPTO_MD5)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the MD5 control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_md5_operation

Perform an MD5 hash operation.

Prototype

```
UINT _nx_crypto_method_md5_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs MD5 hash operation. The MD5 control block must have been initialized with *_nx_crypto_method_md5_init()*. The MD5 algorithm to be performed is based on the algorithm specified in the *method* control block.

For the final *NX_CRYPTO_HASH_CALCULATE* operation, the output buffer size must be 16 bytes.

This operation does not keep state information and does not alter the key material in the MD5 control block.

Parameters

op	Type of operation to perform. Valid operation is: *NX_CRYPTO_HASH_INITIALIZE* *NX_CRYPTO_HASH_UPDATE* *NX_CRYPTO_HASH_CALCULATE*
handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid MD5 crypto method. The crypto method used here must be the same used in the *_nx_crypto_method_md5_init().*
input_data	Points to a buffer containing input text data. There are not restrictions on input buffer.
input_data_size	Size of the input data, in bytes.
iv_ptr	This field is not used for MD5.

iv_size This field is not used for MD5.

output_buffer Pointer to the memory area for the generated MD5 hash.

output_buffer_size
Size of the output buffer in bytes. For MD5 the buffer size must be 16 bytes.

crypto_metadata Pointer to the MD5 control block used in *_nx_crypto_method_md5_init()*.

crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For MD5, the metadata size must *sizeof(NX_CRYPTO_MD5)*

packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

nx_crypto_hw_process_callback
This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the MD5 operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid MD5 algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_md5_cleanup

Clean up the MD5 control block.

Prototype

```
UINT _nx_crypto_method_md5_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the MD5 control block after it determines this MD5 session is no longer needed.

Parameters

crypto_metadata Pointer to the MD5 control block used in *_nx_crypto_method_md5_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the MD5 session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_sha1_init

Initializes the SHA1 crypto control block

Prototype

```
UINT _nx_crypto_method_sha1_init(
                                 struct NX_CRYPTO_METHOD_STRUCT *method,
                                 UCHAR *key,
                                 NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                 VOID **handle,
                                 VOID *crypto_metadata,
                                 ULONG crypto_metadata_size)
```

Description

This function initializes the SHA1 control block with the given key string. Once the SHA1 control block is initialized, subsequent SHA1 operation shall be using the same control block.

Application may create multiple SHA1 control blocks, each represents a session. Initializing the SHA1 control block starts a new hash computation session. Re-initializing the SHA1 control block abandons the current session and stars a new one.
.

Parameters

method	Pointer to a valid SHA1 crypto method control block. The following pre-defined crypto methods are available: *crypto_method_sha1*
key	This field is not used for SHA1.
key_size_in_bits	This field is not used for SHA1
handle	This service returns a handle to the caller. The handle is implementation-dependent and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the SHA1control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For SHA1, the metadata size must be *sizeof(NX_CRYPTO_SHA1)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the SHA1control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_sha1_operation

Perform SHA1 hash operation

Prototype

```
UINT _nx_crypto_method_sha1_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs SHA1 hash operation. The SHA1 control block must have been initialized with *_nx_crypto_method_sha1_init()*. The SHA1 algorithm to be performed is based on the algorithm specified in the *method* control block.

For the final *NX_CRYPTO_HASH_CALCULATE* operation, the output buffer size must be 20 bytes.
.

Parameters

op	Type of operation to perform. Valid operation is: *NX_CRYPTO_HASH_INITIALIZE* *NX_CRYPTO_HASH_UPDATE* *NX_CRYPTO_HASH_CALCULATE*
handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid SHA1 crypto method. The crypto method used here must be the same used in the *nx_crypto_method_sha1_init().*
input_data	Points to a buffer containing input text data. There are not restrictions on input buffer.
input_data_size	Size of the input data, in bytes.
iv_ptr	This field is not used for SHA1.
iv_size	This field is not used for SHA1.

output_buffer Pointer to the memory area for the generated SHA1 hash.

output_buffer_size
Size of the output buffer in bytes. For SHA1 the buffer size must be 20 bytes.

crypto_metadata Pointer to the SHA1 control block used in *_nx_crypto_method_sha1_init()*.

crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For SHA1, the metadata size must *sizeof(NX_CRYPTO_SHA1)*

packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

nx_crypto_hw_process_callback
This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the SHA1 operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid SHA1 algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_sha1_cleanup

Clean up the SHA1 control block.

Prototype

```
UINT _nx_crypto_method_sha1_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the SHA1 control block after it determines this SHA1 session is no longer needed.

Parameters

crypto_metadata Pointer to the SHA1 control block used in *_nx_crypto_method_sha1_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the SHA1 session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_sha256_init

Initializes the SHA256 crypto control block

Prototype

```
UINT _nx_crypto_method_sha256_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the SHA256 control block with the given key string. Once the SHA256 control block is initialized, subsequent SHA256 operation shall be using the same control block.

Application may create multiple SHA256 control blocks, each represents a session. Initializing the SHA256 control block starts a new hash computation session. Re-initializing the SHA256 control block abandons the current session and stars a new one.
.

Parameters

method	Pointer to a valid SHA256 crypto method control block. The following pre-defined crypto methods are available: *crypto_method_sha256* *crypto_method_sha224*
key	This field is not used for SHA256.
key_size_in_bits	This field is not used for SHA256
handle	This service returns a handle to the caller. The handle is implementation-dependent and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the SHA256 control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For SHA256, the metadata size must be *sizeof(NX_CRYPTO_SHA256)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the SHA256 control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_sha256_operation

Perform SHA256 hash operation

Prototype

```
UINT _nx_crypto_method_sha256_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs SHA256 hash operation. The SHA256 control block must have been initialized with ***_nx_crypto_method_sha256_init()***. The SHA256 algorithm to be performed is based on the algorithm specified in the *method* control block.

For the final *NX_CRYPTO_HASH_CALCULATE* operation, the output buffer size must be 32 bytes for SHA256, or 28 bytes for SHA224.
.

Parameters

op	Type of operation to perform. Valid operation is: *NX_CRYPTO_HASH_INITIALIZE* *NX_CRYPTO_HASH_UPDATE* *NX_CRYPTO_HASH_CALCULATE*
handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid SHA256 crypto method. The crypto method used here must be the same used in the *_nx_crypto_method_sha256_init().*
input_data	Points to a buffer containing input text data. There are not restrictions on input buffer.
input_data_size	Size of the input data, in bytes.
iv_ptr	This field is not used for SHA256.
iv_size	This field is not used for SHA256.

output_buffer Pointer to the memory area for the generated SHA256 hash.

output_buffer_size
Size of the output buffer in bytes. For SHA256 the buffer size must be 32 bytes. For SHA224 the buffer size must be 28 bytes.

crypto_metadata Pointer to the SHA2 control block used in *_nx_crypto_method_sha2_init()*.

crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For SHA256, the metadata size must *sizeof(NX_CRYPTO_SHA256)*

packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

nx_crypto_hw_process_callback
This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the SHA256 operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid SHA256 algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_sha256_cleanup

Clean up the SHA256 control block.

Prototype

```
UINT _nx_crypto_method_sha256_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the SHA256 control block after it determines this SHA256 session is no longer needed.

Parameters

crypto_metadata Pointer to the SHA256 control block used in *_nx_crypto_method_sha256_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the SHA256 session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

_nx_crypto_method_sha512_init

Initializes the SHA512 crypto control block

Prototype

```
UINT _nx_crypto_method_sha512_init(
                                struct NX_CRYPTO_METHOD_STRUCT *method,
                                UCHAR *key,
                                NX_CRYPTO_KEY_SIZE key_size_in_bits,
                                VOID **handle,
                                VOID *crypto_metadata,
                                ULONG crypto_metadata_size)
```

Description

This function initializes the SHA512 control block with the given key string. Once the SHA512 control block is initialized, subsequent SHA512 operation shall be using the same control block.

Application may create multiple SHA512 control blocks, each represents a session. Initializing the SHA512 control block starts a new hash computation session. Re-initializing the SHA512 control block abandons the current session and stars a new one.
.

Parameters

method	Pointer to a valid SHA512 crypto method control block. The following pre-defined crypto methods are available: *crypto_method_sha512* *crypto_method_sha384*
key	This field is not used for SHA512.
key_size_in_bits	This field is not used for SHA512
handle	This service returns a handle to the caller. The handle is implementation-dependent and is not being used in this implementation. Application shall pass NULL for the handle.
crypto_metadata	Pointer to a valid memory space for the SHA512 control block. The starting address of the memory space must be 4-byte aligned.
crypto_metadata_size	Size, in bytes, of the crypto_metadata area. For SHA512, the metadata size must be *sizeof(NX_CRYPTO_SHA512)*

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successful initialization of the SHA512 control block with the key and key size.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid pointer to the key, or invalid crypto_metadata or crypto_metadata_size, or crypto_metadata is not 4-byte aligned.

Example

_nx_crypto_method_sha512_operation

Perform SHA512 hash operation

Prototype

```
UINT _nx_crypto_method_sha512_operation(UINT op,
                        VOID *handle,
                        struct NX_CRYPTO_METHOD_STRUCT *method,
                        UCHAR *key,
                        NX_CRYPTO_KEY_SIZE key_size_in_bits,
                        UCHAR *input,
                        ULONG input_length_in_byte,
                        UCHAR *iv_ptr,
                        UCHAR *output,
                        ULONG output_length_in_byte,
                        VOID *crypto_metadata,
                        ULONG crypto_metadata_size,
                        VOID *packet_ptr,
                        VOID (*nx_crypto_hw_process_callback)(VOID
                              *packet_ptr, UINT status))
```

Description

This function performs SHA512 hash operation. The SHA512 control block must have been initialized with *_nx_crypto_method_sha512_init()*. The SHA512 algorithm to be performed is based on the algorithm specified in the *method* control block.

For the final *NX_CRYPTO_HASH_CALCULATE* operation, the output buffer size must be 64 bytes for SHA512, or 48 bytes for SHA384.
.

Parameters

op	Type of operation to perform. Valid operation is: *NX_CRYPTO_HASH_INITIALIZE* *NX_CRYPTO_HASH_UPDATE* *NX_CRYPTO_HASH_CALCULATE*
handle	This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.
method	Pointer to the valid SHA512 crypto method. The crypto method used here must be the same used in the *_nx_crypto_method_sha512_init().*
input_data	Points to a buffer containing input text data. There are not restrictions on input buffer.
input_data_size	Size of the input data, in bytes.
iv_ptr	This field is not used for SHA512.
iv_size	This field is not used for SHA512.

output_buffer Pointer to the memory area for the generated SHA512 hash.

output_buffer_size
Size of the output buffer in bytes. For SHA512 the buffer size must be 64 bytes. For SHA384 the buffer size must be 48 bytes.

crypto_metadata Pointer to the SHA512 control block used in *_nx_crypto_method_sha512_init()*.

crypto_metadata_size
Size, in bytes, of the crypto_metadata area. For SHA512, the metadata size must *sizeof(NX_CRYPTO_SHA512)*

packet_ptr This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

nx_crypto_hw_process_callback
This field is not used in the software implementation of NetX Crypto library. Any values passed in are silently ignored.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully executed the SHA512 operation.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.
NX_PTR_ERROR	(0x07)	Invalid input pointer or invalid length.
NX_CRYPTO_INVALID_ALGORITHM	(0x20004)	Invalid SHA512 algorithm being specified.
NX_CRYPTO_INVALID_BUFFER_SIZE	(0x20005)	Invalid output buffer size.

Example

_nx_crypto_method_sha512_cleanup

Clean up the SHA512 control block.

Prototype

```
UINT _nx_crypto_method_sha512_cleanup(VOID* crypto_metadata)
```

Description

Application calls this function to clean up the SHA512 control block after it determines this SHA512 session is no longer needed.

Parameters

crypto_metadata Pointer to the SHA512 control block used in *_nx_crypto_method_sha512_init()*.

Return Values

NX_CRYPTO_SUCCESS	(0x00)	Successfully cleaned up the SHA512 session.
NX_CRYPTO_INVALID_LIBRARY	(0x20001)	The crypto library is in an invalid state and cannot be used.

Example

Appendix
NetX Crypto CAVS Test

```
NetX Crypto CAVS AES Test:          CBCGFSbox128.......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCGFSbox192.......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCGFSbox256.......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCMCT128..........................SUCCESS!
NetX Crypto CAVS AES Test:          CBCMCT192..........................SUCCESS!
NetX Crypto CAVS AES Test:          CBCMCT256..........................SUCCESS!
NetX Crypto CAVS AES Test:          CBCMMT128..........................SUCCESS!
NetX Crypto CAVS AES Test:          CBCMMT192..........................SUCCESS!
NetX Crypto CAVS AES Test:          CBCMMT256..........................SUCCESS!
NetX Crypto CAVS AES Test:          CBCKeySbox128......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCKeySbox192......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCKeySbox256......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCVarKey128.......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCVarKey192.......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCVarKey256.......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCVarTxt128.......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCVarTxt192.......................SUCCESS!
NetX Crypto CAVS AES Test:          CBCVarTxt256.......................SUCCESS!
NetX Crypto CAVS 3DES Test:         TCBCinvperm 192....................SUCCESS!
NetX Crypto CAVS 3DES Test:         TCBCpermop 192.....................SUCCESS!
NetX Crypto CAVS 3DES Test:         TCBCvarkey 192.....................SUCCESS!
NetX Crypto CAVS 3DES Test:         TCBCvartext 192....................SUCCESS!
NetX Crypto CAVS 3DES Test:         TCBCsubtab 192.....................SUCCESS!
NetX Crypto CAVS 3DES Test:         TCBCMMT2 192.......................SUCCESS!
NetX Crypto CAVS 3DES Test:         TCBCMMT3 192.......................SUCCESS!
NetX Crypto CAVS 3DES Test:         TCBCMonte2 192.....................SUCCESS!
NetX Crypto CAVS 3DES Test:         TCBCMonte3 192.....................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA1LongMsg........................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA1Monte..........................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA1ShortMsg.......................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA224LongMsg......................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA224Monte........................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA224ShortMsg.....................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA256LongMsg......................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA256Monte........................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA256ShortMsg.....................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA384LongMsg......................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA384Monte........................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA384ShortMsg.....................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA512LongMsg......................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA512Monte........................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA512ShortMsg.....................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA512_224LongMsg..................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA512_224Monte....................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA512_224ShortMsg.................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA512_256LongMsg..................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA512_256Monte....................SUCCESS!
NetX Crypto CAVS SHA Test:          SHA512_256ShortMsg.................SUCCESS!
NetX Crypto CAVS HMAC Test:         HMAC...............................SUCCESS!
NetX Crypto CAVS DRBG Test:         CTR_DRBG...........................SUCCESS!
NetX Crypto CAVS KDF Test:          tls................................SUCCESS!
NetX Crypto CAVS RSA2_fixed Test:   SigGen15_186-3.....................SUCCESS!
NetX Crypto CAVS RSA2 Test:         SigGen15_186-3.....................SUCCESS!
NetX Crypto CAVS RSA2_fixed Test:   SigVer15_186-3.....................SUCCESS!
NetX Crypto CAVS ECDSA Test:        KeyPair............................SUCCESS!
NetX Crypto CAVS ECDSA Test:        SigGen.............................SUCCESS!
NetX Crypto CAVS ECDSA Test:        SigGenComponent....................SUCCESS!
NetX Crypto CAVS ECDSA Test:        SigVer.............................SUCCESS!
**** Testing Complete ****
Test finished.
```