

NcApi

Generated by Doxygen 1.8.10

Tue Mar 6 2018 14:01:41

Contents

1	Introduction and overview	1
2	Initialisation and Application provided functions	7
3	Data Structure Index	9
3.1	Data Structures	9
4	File Index	11
4.1	File List	11
5	Data Structure Documentation	13
5.1	NcApi Struct Reference	13
5.2	NcApiAltCmdMessage Struct Reference	13
5.3	NcApiAltCmdParams Struct Reference	14
5.4	NcApiHostAckNack Struct Reference	14
5.5	NcApiHostData Struct Reference	14
5.6	NcApiHostDataHapa Struct Reference	15
5.7	NcApiHostUappData Struct Reference	15
5.8	NcApiHostUappDataHapa Struct Reference	15
5.9	NcApiNeighbor Struct Reference	16
5.10	NcApiNeighborListReply Struct Reference	16
5.11	NcApiNeighborListRequestMessage Struct Reference	16
5.12	NcApiNeighborListRequestParams Struct Reference	17
5.13	NcApiNetCmdMessage Struct Reference	17
5.14	NcApiNetCmdParams Struct Reference	17
5.15	NcApiNetCmdReply Struct Reference	18
5.16	NcApiNodeInfoParams Struct Reference	18
5.17	NcApiNodeInfoReply Struct Reference	19
5.18	NcApiNodeInfoRequestMessage Struct Reference	19
5.19	NcApiRxHandlers Struct Reference	19
5.20	NcApiSendAckMessage Struct Reference	20
5.21	NcApiSendAckParams Struct Reference	20
5.22	NcApiSendUnackMessage Struct Reference	21

5.23	NcApiSendUnackParams Struct Reference	21
5.24	NcApiWesCmdMessage Struct Reference	22
5.25	NcApiWesCmdParams Struct Reference	22
5.26	NcApiWesResponseMessage Struct Reference	22
5.27	NcApiWesResponseParams Struct Reference	23
5.28	NcApiWesSetupRequest Struct Reference	23
5.29	NcApiWesStatus Struct Reference	23
6	File Documentation	25
6.1	DoxygenMainPage.h File Reference	25
6.2	DoxygenPage001.h File Reference	25
6.3	NcApi.h File Reference	25
6.3.1	Typedef Documentation	30
6.3.1.1	pfnNcApiHostAckCallback	30
6.3.1.2	pfnNcApiHostDataCallback	30
6.3.1.3	pfnNcApiHostDataHapaCallback	30
6.3.1.4	pfnNcApiHostUappDataCallback	30
6.3.1.5	pfnNcApiHostUappDataHapaCallback	31
6.3.1.6	pfnNcApiNeighborListReplyCallback	31
6.3.1.7	pfnNcApiNetCmdResponseCallback	31
6.3.1.8	pfnNcApiNodeInfoReplyCallback	31
6.3.1.9	pfnNcApiReadCallback	31
6.3.1.10	pfnNcApiWesSetupRequestCallback	31
6.3.1.11	pfnNcApiWesStatusCallback	32
6.3.1.12	tNcApi	32
6.3.1.13	tNcApiAltCmdMessage	32
6.3.1.14	tNcApiAltCmdParams	32
6.3.1.15	tNcApiHostAckNack	32
6.3.1.16	tNcApiHostData	32
6.3.1.17	tNcApiHostDataHapa	32
6.3.1.18	tNcApiHostUappData	32
6.3.1.19	tNcApiHostUappDataHapa	32
6.3.1.20	tNcApiNeighbor	32
6.3.1.21	tNcApiNeighborListReply	32
6.3.1.22	tNcApiNeighborListRequestMessage	32
6.3.1.23	tNcApiNeighborListRequestParams	32
6.3.1.24	tNcApiNetCmdMessage	32
6.3.1.25	tNcApiNetCmdParams	32
6.3.1.26	tNcApiNetCmdReply	32
6.3.1.27	tNcApiNodeInfoParams	32

6.3.1.28	tNcApiNodeInfoReply	32
6.3.1.29	tNcApiNodeInfoRequestMessage	32
6.3.1.30	tNcApiRxHandlers	32
6.3.1.31	tNcApiSendAckMessage	32
6.3.1.32	tNcApiSendAckParams	32
6.3.1.33	tNcApiSendUnackMessage	32
6.3.1.34	tNcApiSendUnackParams	33
6.3.1.35	tNcApiWesCmdMessage	33
6.3.1.36	tNcApiWesCmdParams	33
6.3.1.37	tNcApiWesResponseMessage	33
6.3.1.38	tNcApiWesResponseParams	33
6.3.1.39	tNcApiWesSetupRequest	33
6.3.1.40	tNcApiWesStatus	33
6.3.2	Enumeration Type Documentation	33
6.3.2.1	NcApiAltCmdValues	33
6.3.2.2	NcApiErrorCodes	33
6.3.2.3	NcApiNetCmdValues	33
6.3.2.4	NcApiNodeType	33
6.3.2.5	NcApiWesCmdValues	34
6.3.2.6	NcApiWesStatusValues	34
6.3.3	Function Documentation	34
6.3.3.1	NcApiCallbackNwuActive(uint8_t n)	34
6.3.3.2	NcApiCancelEnqueuedMessage(uint8_t n)	34
6.3.3.3	NcApiCtsActive(uint8_t n)	34
6.3.3.4	NcApiExecuteCallbacks(uint8_t n, uint8_t *msg, uint8_t msgLength)	34
6.3.3.5	NcApiInit()	35
6.3.3.6	NcApiRxData(uint8_t n, uint8_t byte)	35
6.3.3.7	NcApiSendAcknowledged(uint8_t n, tNcApiSendAckParams *args)	35
6.3.3.8	NcApiSendAltCmd(uint8_t n, tNcApiAltCmdParams *args)	35
6.3.3.9	NcApiSendNeighborListRequest(uint8_t n, void *callbackToken)	35
6.3.3.10	NcApiSendNetCmd(uint8_t n, tNcApiNetCmdParams *args)	36
6.3.3.11	NcApiSendNodeInfoRequest(uint8_t n, tNcApiNodeInfoParams *args)	36
6.3.3.12	NcApiSendRaw(uint8_t n, tNcApiSendAckParams *args)	36
6.3.3.13	NcApiSendUnacknowledged(uint8_t n, tNcApiSendUnackParams *args)	36
6.3.3.14	NcApiSendWesCmd(uint8_t n, tNcApiWesCmdParams *args)	36
6.3.3.15	NcApiSendWesResponse(uint8_t n, tNcApiWesResponseParams *args)	37
6.3.3.16	NcApiSupportMessageReceived(uint8_t n, void *callbackToken, uint8_t *msg, uint8_t msgLength)	37
6.3.3.17	NcApiSupportMessageWritten(uint8_t n, void *callbackToken, uint8_t *finalMsg, uint8_t finalMsgLength)	37

6.3.3.18	NcApiSupportTxData(uint8_t n, uint8_t *finalMsg, uint8_t finalMsgLength)	37
6.3.4	Variable Documentation	38
6.3.4.1	g_ncApi	38
6.3.4.2	g_numberOfNcApis	38
6.4	NeoParser.h File Reference	38
6.4.1	Typedef Documentation	40
6.4.1.1	NcApiMessageType	40
6.4.2	Enumeration Type Documentation	40
6.4.2.1	NcApiMessageType	40
6.4.3	Function Documentation	40
6.4.3.1	NcApiGetMsgAsHostAck(uint8_t *buffer, tNcApiHostAckNack *p)	40
6.4.3.2	NcApiGetMsgAsHostData(uint8_t *buffer, tNcApiHostData *p)	40
6.4.3.3	NcApiGetMsgAsHostDataHapa(uint8_t *buffer, tNcApiHostDataHapa *p)	40
6.4.3.4	NcApiGetMsgAsHostUappData(uint8_t *buffer, tNcApiHostUappData *p)	41
6.4.3.5	NcApiGetMsgAsHostUappDataHapa(uint8_t *buffer, tNcApiHostUappDataHapa *p)	41
6.4.3.6	NcApiGetMsgAsNeighborListReply(uint8_t *buffer, tNcApiNeighborListReply *p)	41
6.4.3.7	NcApiGetMsgAsNetCmdResponse(uint8_t *buffer, tNcApiNetCmdReply *p)	41
6.4.3.8	NcApiGetMsgAsNodeInfo(uint8_t *buffer, tNcApiNodeInfoReply *p)	41
6.4.3.9	NcApiGetMsgAsNodeInfoReply(uint8_t *buffer, tNcApiNodeInfoReply *p)	41
6.4.3.10	NcApiGetMsgAsWesSetupRequest(uint8_t *buffer, tNcApiWesSetupRequest *p)	41
6.4.3.11	NcApiGetMsgAsWesStatus(uint8_t *buffer, tNcApiWesStatus *p)	42
6.4.3.12	NcApiIsMsgGetBootLoaderVersionReply(uint8_t *buffer, uint16_t startAt, uint16_t length)	42
6.4.3.13	NcApiIsMsgGetProtocolVersionReply(uint8_t *buffer, uint16_t startAt, uint16_t length)	42
6.4.3.14	NcApiIsMsgReadOtpReply(uint8_t *buffer, uint16_t startAt, uint16_t length)	42
6.4.3.15	NcApiIsValidApiFrame(uint8_t *buffer, uint16_t bufLength, uint16_t *outStartAt, uint16_t *outLength)	42
6.4.3.16	NcApiIsValidFrameTraceSpecific(uint8_t *buffer, uint16_t bufLength)	43
6.4.3.17	NcApiIsValidSysFrame(uint8_t *buffer, uint16_t bufLength, uint16_t *outStartAt, uint16_t *outLength)	43
	Index	45

Chapter 1

Introduction and overview

The application layer interfaces with the node through the API when payload data is being transmitted or received. The API implements handles for encoding/decoding and synchronisation of the data transfer to and from the node.

For further information, please also refer to the documents on NeoCortec's download page:

neocortec.com/downloads/

Overview of the API context

The Application using the API typically resides in a microcontroller which communicates with the NeoCortec node through a UART.

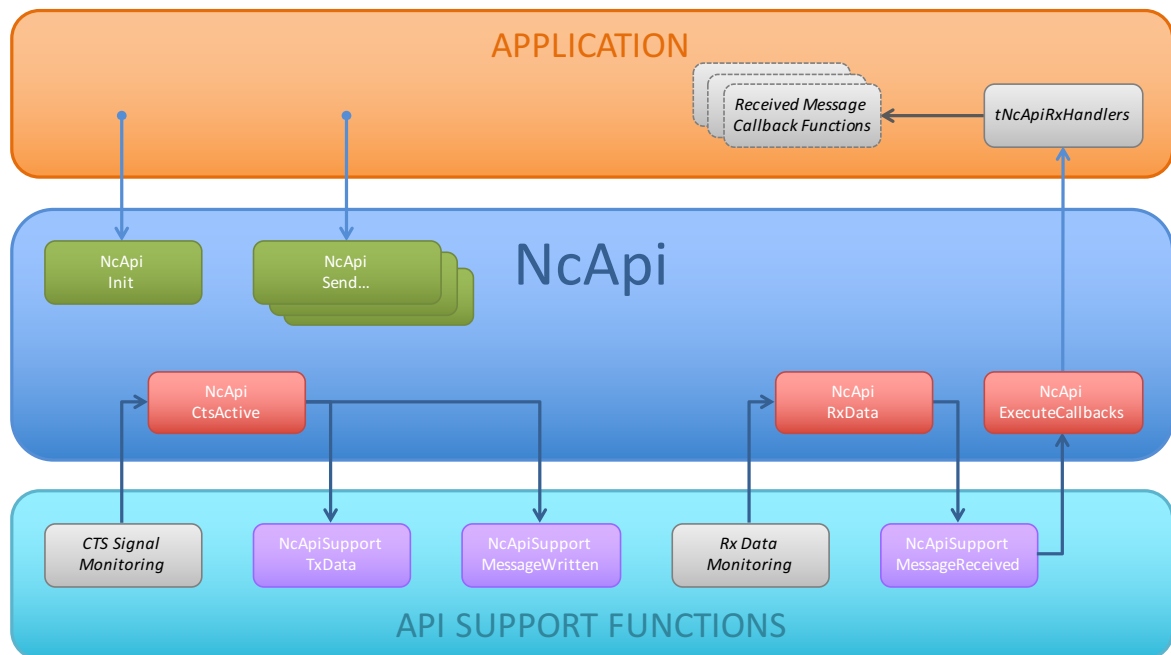
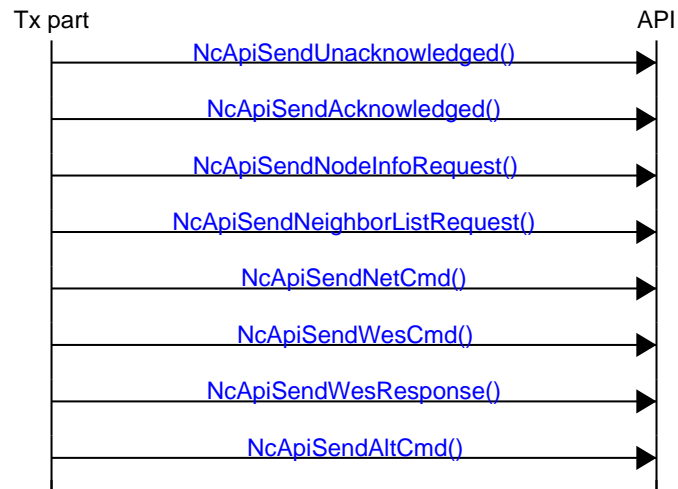


Figure 1.1: API context

Interface between the API and the Tx part of the Application

Functions for initiating a message to the NeoCortec module by enqueueing it for transmission:

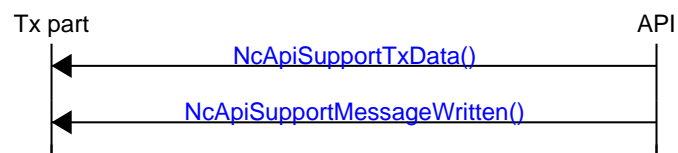


Unfortunately the links in the sequence diagram does not work in the PDF version of the documentation, and hence links to the functions are provided here:

- [NcApiSendUnacknowledged\(\)](#)
- [NcApiSendAcknowledged\(\)](#)
- [NcApiSendNodeInfoRequest\(\)](#)
- [NcApiSendNeighborListRequest\(\)](#)
- [NcApiSendNetCmd\(\)](#)
- [NcApiSendWesCmd\(\)](#)
- [NcApiSendWesResponse\(\)](#)
- [NcApiSendAltCmd\(\)](#)

Interface between the API and the Tx part of the API Support

Functions related to the actual transmission of a message to the NeoCortec module as response to the CTS signal:



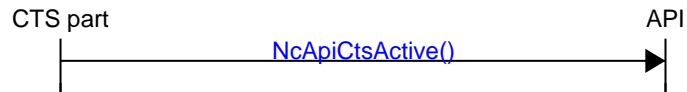
[NcApiSupportTxData\(\)](#) will be called if there is a pending enqueued message when [NcApiCtsActive\(\)](#) is called. Please also refer to the section "Interface between the API and the CTS part of the API Support".

Unfortunately the links in the sequence diagram does not work in the PDF version of the documentation, and hence links to the functions are provided here:

- [NcApiSupportTxData\(\)](#) (provided by the API Support)
- [NcApiSupportMessageWritten\(\)](#)

Interface between the API and the CTS part of the API Support

Function to inform the API that the NeoCortec module is ready to receive messages:



If there is a pending enqueued message the API will call [NcApiSupportTxData\(\)](#), and when the entire message has been delivered to the UART, [NcApiSupportMessageWritten\(\)](#) is called.

Please also refer to the section "Interface between the API and the Tx part of the API Support".

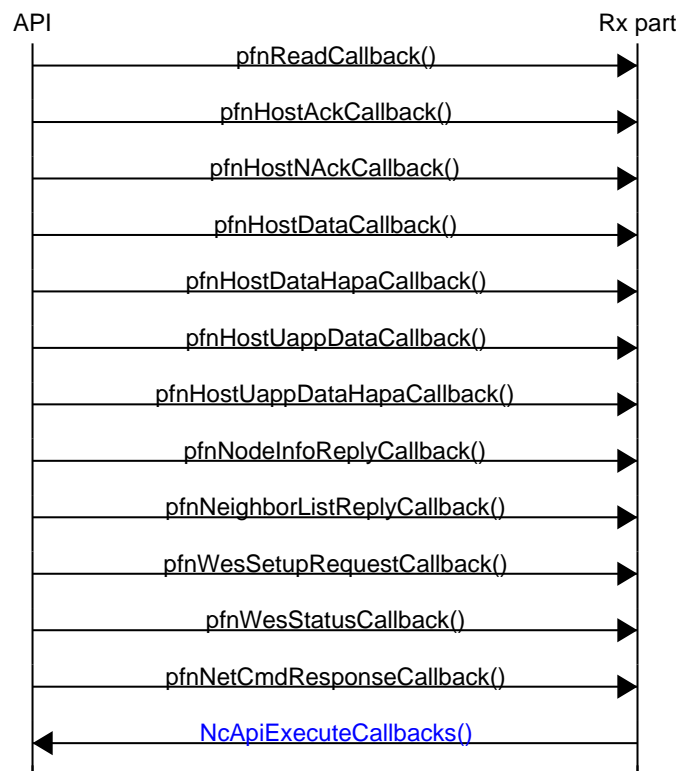
Unfortunately the links in the sequence diagram does not work in the PDF version of the documentation, and hence links to the functions are provided here:

- [NcApiCtsActive\(\)](#)

Interface between the API and the Rx part of the Application

When data is received from the mesh network, a series of call-back functions will call the Application with the received data. Depending on what type of data is received, a type specific call-back is issued. It is optional for the application layer to register for the call-backs.

Call back functions to pass on received messages to the application:

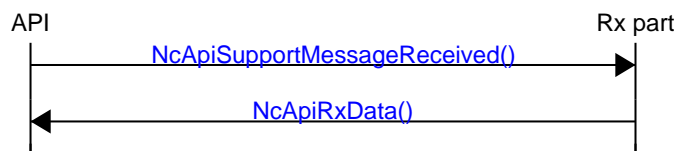


Unfortunately the links in the sequence diagram does not work in the PDF version of the documentation, and hence links to the functions are provided here:

- [pfn...Callback\(\)](#) (provided by the Application)
- [NcApiExecuteCallbacks\(\)](#)

Interface between the API and the Rx part of the API Support

Functions related to reception of messages from the NeoCortec module:



When data is received from the UART, it shall be forwarded to [NcApiRxData\(\)](#). When a complete message has been received, the application will be notified via [NcApiSupportMessageReceived\(\)](#).

Unfortunately the links in the sequence diagram does not work in the PDF version of the documentation, and hence links to the functions are provided here:

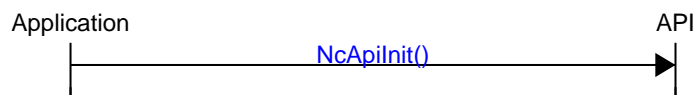
- [NcApiSupportMessageReceived\(\)](#) (provided by the API Support)
[NcApiRxData\(\)](#)

Chapter 2

Initialisation and Application provided functions

Initialisation

The application must call `NcApiInit()` once for initialisation.



UART service functions

The Application must provide the following actions to UART service requests, cf. the "Introduction and overview" chapter:

- CTS active low:
`NcApiCtsActive()`
- RX data (optionally also RT):
`NcApiRxData()`

The application must provide means of servicing UART CTS signal changes. When the CTS edges low, `NcApiCtsActive()` must be called. If there is a pending enqueued message the API will call `NcApiSupportTxData()`, and when the entire message has been delivered to the UART, `NcApiSupportMessageWritten()` is called.

The application must provide means of servicing UART Rx data. When a byte is received `NcApiRxData()` must be called. When a complete message has been received, the application will be notified via `NcApiSupportMessageReceived()`.

Basic Rx/Tx functions

The Application must provide the following basic functions for receiving and transmitting messages, cf. the "Introduction and overview" chapter:

- `NcApiSupportTxData()`
- `NcApiSupportMessageWritten()`
- `NcApiSupportMessageReceived()`

Received message callback functions

The applicaton can chose to be notified for certain received message types through the instance of [NcApiRx↔
Handlers](#), which is aet of application callback function pointers.

Cf. the "Introduction and overview" chapter.

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

NcApi	This is the definition of a global structure holding various information, in particular the RX and TX buffers for a specific UART, and the set of application callbacks to handle any received messages. These data are managed by the NcApi module, and as such the fields are considered internal to NcApi	13
NcApiAltCmdMessage	Definition of message type "0x20: ALT command"	13
NcApiAltCmdParams	Parameters for the function handling message type "0x20: ALT command"	14
NcApiHostAckNack	Parameters for the function handling message type "0x50: Acknowledge for previously sent packet" Parameters for the function handling message type "0x51: Non-Acknowledge for previously sent packet"	14
NcApiHostData	Parameters for the function handling message type "0x52: Host Data"	14
NcApiHostDataHapa	Parameters for the function handling message type "0x53: Host Data HAPA"	15
NcApiHostUappData	Parameters for the function handling message type "0x54: Host Data Unacknowledged"	15
NcApiHostUappDataHapa	Parameters for the function handling message type "0x55: Host Data HAPA Unacknowledged"	15
NcApiNeighbor	16
NcApiNeighborListReply	Parameters for the function handling message type "0x59: Neighbor List Reply"	16
NcApiNeighborListRequestMessage	Definition of message type "0x09: Neighbor List Request"	16
NcApiNeighborListRequestParams	Parameters for the function handling message type "0x09: Neighbor List Request"	17
NcApiNetCmdMessage	Definition of message type "0x0a: Network Command"	17
NcApiNetCmdParams	Parameters for the function handling message type "0x0a: Network Command"	17
NcApiNetCmdReply	Parameters for the function handling message type "0x5a: Network command response"	18
NcApiNodeInfoParams	Parameters for the function handling message type "0x08: Node Info Request"	18

NcApiNodeInfoReply	
Parameters for the function handling message type "0x58: Node Info Reply"	19
NcApiNodeInfoRequestMessage	
Definition of message type "0x08: Node Info Request"	19
NcApiRxHandlers	
Set of application callbacks to handle any received messages. Each callback is optional allowing the application to register specific callbacks only for the message types of particular interest . .	19
NcApiSendAckMessage	
Definition of message type "0x03: Acknowledged Packet"	20
NcApiSendAckParams	
Parameters for the function handling message type "0x03: Acknowledged Packet"	20
NcApiSendUnackMessage	
Definition of message type "0x02: Unacknowledged Packet"	21
NcApiSendUnackParams	
Parameters for the function handling message type "0x02: Unacknowledged Packet"	21
NcApiWesCmdMessage	
Definition of message type "0x10: WES Command"	22
NcApiWesCmdParams	
Parameters for the function handling message type "0x10: WES Command"	22
NcApiWesResponseMessage	22
NcApiWesResponseParams	
Parameters for the function handling message type "0x11: WES Setup Response"	23
NcApiWesSetupRequest	
Parameters for the function handling message type "0x61: WES Setup Request"	23
NcApiWesStatus	
Parameters for the function handling message type "0x60: WES Status"	23

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

DoxygenMainPage.h	25
DoxygenPage001.h	25
NcApi.h	25
NeoParser.h	38

Chapter 5

Data Structure Documentation

5.1 NcApi Struct Reference

This is the definition of a global structure holding various information, in particular the RX and TX buffers for a specific UART, and the set of application callbacks to handle any received messages. These data are managed by the [NcApi](#) module, and as such the fields are considered internal to [NcApi](#).

```
#include <NcApi.h>
```

Data Fields

- `uint8_t rxBuffer` [255]
Internal UART receive buffer.
- `uint16_t rxPosition`
Internal position in UART receive buffer.
- `volatile uint8_t txMsgLen`
Internal UART transmit buffer length.
- `uint8_t txBuffer` [32]
Internal UART transmit buffer.
- `void * writeCallbackToken`
Internal UART transmit callback token.
- `volatile uint8_t recvBufIsSynced`
Internal UART receive buffer in sync.
- `tNcApiRxHandlers * NcApiRxHandlers`
Set of application callbacks to handle any received messages.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.2 NcApiAltCmdMessage Struct Reference

Definition of message type "0x20: ALT command".

```
#include <NcApi.h>
```

Data Fields

- [NcApiAltCmdValues cmd](#)

ALT Command.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.3 NcApiAltCmdParams Struct Reference

Parameters for the function handling message type "0x20: ALT command".

```
#include <NcApi.h>
```

Data Fields

- [tNcApiAltCmdMessage msg](#)

The actual corresponding message.

- void * [callbackToken](#)

Application provided token / context / tag that it wants to called back with. [NcApi](#) does not inspect this parameter, it merely passes it along.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.4 NcApiHostAckNack Struct Reference

Parameters for the function handling message type "0x50: Acknowledge for previously sent packet"

Parameters for the function handling message type "0x51: Non-Acknowledge for previously sent packet".

```
#include <NcApi.h>
```

Data Fields

- uint16_t [originId](#)

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.5 NcApiHostData Struct Reference

Parameters for the function handling message type "0x52: Host Data".

```
#include <NcApi.h>
```

Data Fields

- uint16_t [originId](#)
- uint16_t [packageAge](#)
- uint8_t [port](#)
- uint8_t [payloadLength](#)
- uint8_t * [payload](#)

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.6 NcApiHostDataHapa Struct Reference

Parameters for the function handling message type "0x53: Host Data HAPA".

```
#include <NcApi.h>
```

Data Fields

- uint16_t [originId](#)
- uint32_t [packageAge](#)
- uint8_t [port](#)
- uint8_t [payloadLength](#)
- uint8_t * [payload](#)

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.7 NcApiHostUappData Struct Reference

Parameters for the function handling message type "0x54: Host Data Unacknowledged".

```
#include <NcApi.h>
```

Data Fields

- uint16_t [originId](#)
- uint16_t [packageAge](#)
- uint8_t [port](#)
- uint16_t [appSeqNo](#)
- uint8_t [payloadLength](#)
- uint8_t * [payload](#)

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.8 NcApiHostUappDataHapa Struct Reference

Parameters for the function handling message type "0x55: Host Data HAPA Unacknowledged".

```
#include <NcApi.h>
```

Data Fields

- uint16_t [originId](#)
- uint32_t [packageAge](#)
- uint8_t [port](#)
- uint16_t [appSeqNo](#)
- uint8_t [payloadLength](#)
- uint8_t * [payload](#)

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.9 NcApiNeighbor Struct Reference

```
#include <NcApi.h>
```

Data Fields

- uint16_t [nodeId](#)
- uint8_t [RSSI](#)

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.10 NcApiNeighborListReply Struct Reference

Parameters for the function handling message type "0x59: Neighbor List Reply".

```
#include <NcApi.h>
```

Data Fields

- uint8_t [NeighborsCount](#)
Numbers of neighbors.
- [tNcApiNeighbor Neighbor](#) [12]
Array of neighbors.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.11 NcApiNeighborListRequestMessage Struct Reference

Definition of message type "0x09: Neighbor List Request".

```
#include <NcApi.h>
```

Data Fields

- void * [dummy](#)
(No parameters)

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.12 NcApiNeighborListRequestParams Struct Reference

Parameters for the function handling message type "0x09: Neighbor List Request".

```
#include <NcApi.h>
```

Data Fields

- [tNcApiNeighborListRequestMessage msg](#)
The actual corresponding message.
- void * [callbackToken](#)
Application provided token / context / tag that it wants to called back with. [NcApi](#) does not inspect this parameter, it merely passes it along.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.13 NcApiNetCmdMessage Struct Reference

Definition of message type "0x0a: Network Command".

```
#include <NcApi.h>
```

Data Fields

- uint16_t [destNodeId](#)
Destination node ID.
- uint8_t [cmd](#)
Network Command.
- uint8_t * [payload](#)
Pointer to payload, if any.
- uint8_t [payloadLength](#)
PayloadLength Length of payload.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.14 NcApiNetCmdParams Struct Reference

Parameters for the function handling message type "0x0a: Network Command".

```
#include <NcApi.h>
```

Data Fields

- [tNcApiNetCmdMessage msg](#)

The actual corresponding message.

- void * [callbackToken](#)

Application provided token / context / tag that it wants to called back with. [NcApi](#) does not inspect this parameter, it merely passes it along.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.15 NcApiNetCmdReply Struct Reference

Parameters for the function handling message type "0x5a: Network command response".

```
#include <NcApi.h>
```

Data Fields

- uint16_t [originId](#)

- [NcApiNetCmdValues cmd](#)

- uint8_t [payloadLength](#)

PayloadLength Length of payload.

- uint8_t * [payload](#)

Pointer to payload, if any.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.16 NcApiNodeInfoParams Struct Reference

Parameters for the function handling message type "0x08: Node Info Request".

```
#include <NcApi.h>
```

Data Fields

- [tNcApiNodeInfoRequestMessage msg](#)

The actual corresponding message.

- void * [callbackToken](#)

Application provided token / context / tag that it wants to called back with. [NcApi](#) does not inspect this parameter, it merely passes it along.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.17 NcApiNodeInfoReply Struct Reference

Parameters for the function handling message type "0x58: Node Info Reply".

```
#include <NcApi.h>
```

Data Fields

- [uint16_t nodeId](#)
Node ID.
- [uint8_t uid](#) [5]
Node uid.
- [NcApiNodeType](#) Type
Node Hardware Type.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.18 NcApiNodeInfoRequestMessage Struct Reference

Definition of message type "0x08: Node Info Request".

```
#include <NcApi.h>
```

Data Fields

- [void *](#) [dummy](#)
(No parameters)

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.19 NcApiRxHandlers Struct Reference

Set of application callbacks to handle any received messages. Each callback is optional allowing the application to register specific callbacks only for the message types of particular interest.

```
#include <NcApi.h>
```

Data Fields

- [pfnNcApiReadCallback](#) [pfnReadCallback](#)
Optional callback for all received messages as a byte array.
- [pfnNcApiHostAckCallback](#) [pfnHostAckCallback](#)
Optional callback for all received HostAck messages.
- [pfnNcApiHostAckCallback](#) [pfnHostNAckCallback](#)
Optional callback for all received HostNAck messages.
- [pfnNcApiHostDataCallback](#) [pfnHostDataCallback](#)
Optional callback for all received HostData messages.

- [pfnNcApiHostDataHapaCallback](#) [pfnHostDataHapaCallback](#)
Optional callback for all received HostDataHapa messages.
- [pfnNcApiHostUappDataCallback](#) [pfnHostUappDataCallback](#)
Optional callback for all received HostUappData messages.
- [pfnNcApiHostUappDataHapaCallback](#) [pfnHostUappDataHapaCallback](#)
Optional callback for all received HostUappDataHapa messages.
- [pfnNcApiNodeInfoReplyCallback](#) [pfnNodeInfoReplyCallback](#)
Optional callback for all received NodeInfoReply messages.
- [pfnNcApiNeighborListReplyCallback](#) [pfnNeighborListReplyCallback](#)
Optional callback for all received NeighborListReply messages.
- [pfnNcApiNetCmdResponseCallback](#) [pfnNetCmdResponseCallback](#)
Optional callback for all received NeighborListReply messages.
- [pfnNcApiWesSetupRequestCallback](#) [pfnWesSetupRequestCallback](#)
Optional callback for all received WesSetupRequest messages.
- [pfnNcApiWesStatusCallback](#) [pfnWesStatusCallback](#)
Optional callback for all received WesStatus messages.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.20 NcApiSendAckMessage Struct Reference

Definition of message type "0x03: Acknowledged Packet".

```
#include <NcApi.h>
```

Data Fields

- `uint16_t` [destNodeId](#)
Destination node ID.
- `uint8_t` [destPort](#)
Destination port.
- `uint8_t` [payloadLength](#)
PayloadLength Length of payload.
- `uint8_t *` [payload](#)
Pointer to payload, if any.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.21 NcApiSendAckParams Struct Reference

Parameters for the function handling message type "0x03: Acknowledged Packet".

```
#include <NcApi.h>
```

Data Fields

- [tNcApiSendAckMessage msg](#)
The actual corresponding message.
- void * [callbackToken](#)
Application provided token / context / tag that it wants to called back with. [NcApi](#) does not inspect this parameter, it merely passes it along.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.22 NcApiSendUnackMessage Struct Reference

Definition of message type "0x02: Unacknowledged Packet".

```
#include <NcApi.h>
```

Data Fields

- uint16_t [destNodeId](#)
Destination node ID.
- uint8_t [destPort](#)
Destination port.
- uint16_t [appSeqNo](#)
Application sequence number.
- uint8_t * [payload](#)
Pointer to payload, if any.
- uint8_t [payloadLength](#)
PayloadLength Length of payload.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.23 NcApiSendUnackParams Struct Reference

Parameters for the function handling message type "0x02: Unacknowledged Packet".

```
#include <NcApi.h>
```

Data Fields

- [tNcApiSendUnackMessage msg](#)
The actual corresponding message.
- void * [callbackToken](#)
Application provided token / context / tag that it wants to called back with. [NcApi](#) does not inspect this parameter, it merely passes it along.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.24 NcApiWesCmdMessage Struct Reference

Definition of message type "0x10: WES Command".

```
#include <NcApi.h>
```

Data Fields

- [NcApiWesCmdValues cmd](#)
WES Command.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.25 NcApiWesCmdParams Struct Reference

Parameters for the function handling message type "0x10: WES Command".

```
#include <NcApi.h>
```

Data Fields

- [tNcApiWesCmdMessage msg](#)
The actual corresponding message.
- void * [callbackToken](#)
Application provided token / context / tag that it wants to called back with. [NcApi](#) does not inspect this parameter, it merely passes it along.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.26 NcApiWesResponseMessage Struct Reference

```
#include <NcApi.h>
```

Data Fields

- uint8_t [uid](#) [5]
UID.
- uint16_t [nodeId](#)
NodeId.
- uint8_t [appSettings](#) [24]
appSettings

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.27 NcApiWesResponseParams Struct Reference

Parameters for the function handling message type "0x11: WES Setup Response".

```
#include <NcApi.h>
```

Data Fields

- [tNcApiWesResponseMessage msg](#)
The actual corresponding message.
- void * [callbackToken](#)
Application provided token / context / tag that it wants to called back with. [NcApi](#) does not inspect this parameter, it merely passes it along.

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.28 NcApiWesSetupRequest Struct Reference

Parameters for the function handling message type "0x61: WES Setup Request".

```
#include <NcApi.h>
```

Data Fields

- uint8_t [uid](#) [5]
- uint8_t [appFuncType](#)

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

5.29 NcApiWesStatus Struct Reference

Parameters for the function handling message type "0x60: WES Status".

```
#include <NcApi.h>
```

Data Fields

- uint8_t [Status](#)
See [NcApiWesStatusValues](#).

The documentation for this struct was generated from the following file:

- [NcApi.h](#)

Chapter 6

File Documentation

6.1 DoxygenMainPage.h File Reference

6.2 DoxygenPage001.h File Reference

6.3 NcApi.h File Reference

```
#include <stdint.h>
```

Data Structures

- struct [NcApiNeighbor](#)
- struct [NcApiSendUnackMessage](#)
Definition of message type "0x02: Unacknowledged Packet".
- struct [NcApiSendUnackParams](#)
Parameters for the function handling message type "0x02: Unacknowledged Packet".
- struct [NcApiSendAckMessage](#)
Definition of message type "0x03: Acknowledged Packet".
- struct [NcApiSendAckParams](#)
Parameters for the function handling message type "0x03: Acknowledged Packet".
- struct [NcApiNodeInfoRequestMessage](#)
Definition of message type "0x08: Node Info Request".
- struct [NcApiNodeInfoParams](#)
Parameters for the function handling message type "0x08: Node Info Request".
- struct [NcApiNeighborListRequestMessage](#)
Definition of message type "0x09: Neighbor List Request".
- struct [NcApiNeighborListRequestParams](#)
Parameters for the function handling message type "0x09: Neighbor List Request".
- struct [NcApiNetCmdMessage](#)
Definition of message type "0x0a: Network Command".
- struct [NcApiNetCmdParams](#)
Parameters for the function handling message type "0x0a: Network Command".
- struct [NcApiWesCmdMessage](#)
Definition of message type "0x10: WES Command".
- struct [NcApiWesCmdParams](#)

Parameters for the function handling message type "0x10: WES Command".

- struct [NcApiWesResponseMessage](#)
- struct [NcApiWesResponseParams](#)

Parameters for the function handling message type "0x11: WES Setup Response".

- struct [NcApiAltCmdMessage](#)

Definition of message type "0x20: ALT command".

- struct [NcApiAltCmdParams](#)

Parameters for the function handling message type "0x20: ALT command".

- struct [NcApiHostAckNack](#)

Parameters for the function handling message type "0x50: Acknowledge for previously sent packet"

Parameters for the function handling message type "0x51: Non-Acknowledge for previously sent packet".

- struct [NcApiHostData](#)

Parameters for the function handling message type "0x52: Host Data".

- struct [NcApiHostDataHapa](#)

Parameters for the function handling message type "0x53: Host Data HAPA".

- struct [NcApiHostUappData](#)

Parameters for the function handling message type "0x54: Host Data Unacknowledged".

- struct [NcApiHostUappDataHapa](#)

Parameters for the function handling message type "0x55: Host Data HAPA Unacknowledged".

- struct [NcApiNodeInfoReply](#)

Parameters for the function handling message type "0x58: Node Info Reply".

- struct [NcApiNeighborListReply](#)

Parameters for the function handling message type "0x59: Neighbor List Reply".

- struct [NcApiNetCmdReply](#)

Parameters for the function handling message type "0x5a: Network command response".

- struct [NcApiWesStatus](#)

Parameters for the function handling message type "0x60: WES Status".

- struct [NcApiWesSetupRequest](#)

Parameters for the function handling message type "0x61: WES Setup Request".

- struct [NcApiRxHandlers](#)

Set of application callbacks to handle any received messages. Each callback is optional allowing the application to register specific callbacks only for the message types of particular interest.

- struct [NcApi](#)

This is the definition of a global structure holding various information, in particular the RX and TX buffers for a specific UART, and the set of application callbacks to handle any received messages. These data are managed by the [NcApi](#) module, and as such the fields are considered internal to [NcApi](#).

Macros

- #define [NCAPI_TXBUFFER_SIZE](#) 32

Default TX buffer size. Can be defined by the application.

- #define [NCAPI_MAX_PAYLOAD_LENGTH](#) ([NCAPI_TXBUFFER_SIZE](#)-5)

- #define [NCAPI_RXBUFFER_SIZE](#) 255

Default RX buffer size. Can be defined by the application.

- #define [WES_APPSETTINGS_LENGTH](#) 24

Definition of message type "0x11: WES Setup Response".

Typedefs

- typedef struct [NcApiNeighbor](#) [tNcApiNeighbor](#)
- typedef struct [NcApiSendUnackMessage](#) [tNcApiSendUnackMessage](#)
Definition of message type "0x02: Unacknowledged Packet".
- typedef struct [NcApiSendUnackParams](#) [tNcApiSendUnackParams](#)
Parameters for the function handling message type "0x02: Unacknowledged Packet".
- typedef struct [NcApiSendAckMessage](#) [tNcApiSendAckMessage](#)
Definition of message type "0x03: Acknowledged Packet".
- typedef struct [NcApiSendAckParams](#) [tNcApiSendAckParams](#)
Parameters for the function handling message type "0x03: Acknowledged Packet".
- typedef struct [NcApiNodeInfoRequestMessage](#) [tNcApiNodeInfoRequestMessage](#)
Definition of message type "0x08: Node Info Request".
- typedef struct [NcApiNodeInfoParams](#) [tNcApiNodeInfoParams](#)
Parameters for the function handling message type "0x08: Node Info Request".
- typedef struct [NcApiNeighborListRequestMessage](#) [tNcApiNeighborListRequestMessage](#)
Definition of message type "0x09: Neighbor List Request".
- typedef struct [NcApiNeighborListRequestParams](#) [tNcApiNeighborListRequestParams](#)
Parameters for the function handling message type "0x09: Neighbor List Request".
- typedef struct [NcApiNetCmdMessage](#) [tNcApiNetCmdMessage](#)
Definition of message type "0x0a: Network Command".
- typedef struct [NcApiNetCmdParams](#) [tNcApiNetCmdParams](#)
Parameters for the function handling message type "0x0a: Network Command".
- typedef struct [NcApiWesCmdMessage](#) [tNcApiWesCmdMessage](#)
Definition of message type "0x10: WES Command".
- typedef struct [NcApiWesCmdParams](#) [tNcApiWesCmdParams](#)
Parameters for the function handling message type "0x10: WES Command".
- typedef struct [NcApiWesResponseMessage](#) [tNcApiWesResponseMessage](#)
- typedef struct [NcApiWesResponseParams](#) [tNcApiWesResponseParams](#)
Parameters for the function handling message type "0x11: WES Setup Response".
- typedef struct [NcApiAltCmdMessage](#) [tNcApiAltCmdMessage](#)
Definition of message type "0x20: ALT command".
- typedef struct [NcApiAltCmdParams](#) [tNcApiAltCmdParams](#)
Parameters for the function handling message type "0x20: ALT command".
- typedef struct [NcApiHostAckNack](#) [tNcApiHostAckNack](#)
Parameters for the function handling message type "0x50: Acknowledge for previously sent packet"
Parameters for the function handling message type "0x51: Non-Acknowledge for previously sent packet".
- typedef struct [NcApiHostData](#) [tNcApiHostData](#)
Parameters for the function handling message type "0x52: Host Data".
- typedef struct [NcApiHostDataHapa](#) [tNcApiHostDataHapa](#)
Parameters for the function handling message type "0x53: Host Data HAPA".
- typedef struct [NcApiHostUappData](#) [tNcApiHostUappData](#)
Parameters for the function handling message type "0x54: Host Data Unacknowledged".
- typedef struct [NcApiHostUappDataHapa](#) [tNcApiHostUappDataHapa](#)
Parameters for the function handling message type "0x55: Host Data HAPA Unacknowledged".
- typedef struct [NcApiNodeInfoReply](#) [tNcApiNodeInfoReply](#)
Parameters for the function handling message type "0x58: Node Info Reply".
- typedef struct [NcApiNeighborListReply](#) [tNcApiNeighborListReply](#)
Parameters for the function handling message type "0x59: Neighbor List Reply".
- typedef struct [NcApiNetCmdReply](#) [tNcApiNetCmdReply](#)
Parameters for the function handling message type "0x5a: Network command response".

- typedef struct [NcApiWesStatus](#) [tNcApiWesStatus](#)
Parameters for the function handling message type "0x60: WES Status".
- typedef struct [NcApiWesSetupRequest](#) [tNcApiWesSetupRequest](#)
Parameters for the function handling message type "0x61: WES Setup Request".
- typedef void(* [pfnNcApiReadCallback](#)) (uint8_t n, uint8_t *msg, uint8_t msgLength)
Application provided function that [NcApi](#) calls whenever any valid NeocCortec messages has been received.
- typedef void(* [pfnNcApiHostAckCallback](#)) (uint8_t n, [tNcApiHostAckNack](#) *m)
Application provided functions that [NcApi](#) calls when a message type "0x50: Acknowledge for previously sent packet" is received, or a message type "0x51: Non-Acknowledge for previously sent packet" is received.
- typedef void(* [pfnNcApiHostDataCallback](#)) (uint8_t n, [tNcApiHostData](#) *m)
Application provided function that [NcApi](#) calls when a message type "0x52: Host Data" is received.
- typedef void(* [pfnNcApiHostDataHapaCallback](#)) (uint8_t n, [tNcApiHostDataHapa](#) *m)
Application provided function that [NcApi](#) calls when a message type "0x53: Host Data HAPA" is received.
- typedef void(* [pfnNcApiHostUappDataCallback](#)) (uint8_t n, [tNcApiHostUappData](#) *m)
Application provided function that [NcApi](#) calls when a message type "0x54: Host Data Unacknowledged" is received.
- typedef void(* [pfnNcApiHostUappDataHapaCallback](#)) (uint8_t n, [tNcApiHostUappDataHapa](#) *m)
Application provided function that [NcApi](#) calls when a message type "0x55: Host Data HAPA Unacknowledged" is received.
- typedef void(* [pfnNcApiNodeInfoReplyCallback](#)) (uint8_t n, [tNcApiNodeInfoReply](#) *m)
Application provided function that [NcApi](#) calls when a message type "0x58: Node Info Reply" is received.
- typedef void(* [pfnNcApiNeighborListReplyCallback](#)) (uint8_t n, [tNcApiNeighborListReply](#) *m)
Application provided function that [NcApi](#) calls when a message type "0x59: Neighbor List Reply" is received.
- typedef void(* [pfnNcApiNetCmdResponseCallback](#)) (uint8_t n, [tNcApiNetCmdReply](#) *m)
Application provided function that [NcApi](#) calls when a message type "0x5a: Network Command Reply" is received.
- typedef void(* [pfnNcApiWesStatusCallback](#)) (uint8_t n, [tNcApiWesStatus](#) *m)
Application provided function that [NcApi](#) calls when a message type "0x60: WES Status" is received.
- typedef void(* [pfnNcApiWesSetupRequestCallback](#)) (uint8_t n, [tNcApiWesSetupRequest](#) *m)
Application provided function that [NcApi](#) calls when a message type "0x61: WES Setup Request" is received.
- typedef struct [NcApiRxHandlers](#) [tNcApiRxHandlers](#)
Set of application callbacks to handle any received messages. Each callback is optional allowing the application to register specific callbacks only for the message types of particular interest.
- typedef struct [NcApi](#) [tNcApi](#)
This is the definition of a global structure holding various information, in particular the RX and TX buffers for a specific UART, and tha set of application callbacks to handle any received messages. These data are managed by the [NcApi](#) module, and as such the fields are considered internal to [NcApi](#).

Enumerations

- enum [NcApiErrorCodes](#) {
[NCAPI_OK](#) = 0, [NCAPI_ERR_NODEID](#) = 1, [NCAPI_ERR_DESTPORT](#) = 2, [NCAPI_ERR_PAYLOAD](#) = 3,
[NCAPI_ERR_ENQUEUED](#) = 4, [NCAPI_ERR_NULLPAYLOAD](#) = 5, [NCAPI_ERR_NOARGS](#) = 6 }
- enum [NcApiWesCmdValues](#) { [NCAPI_WES_STOP](#) = 0, [NCAPI_WES_STARTSERVER](#) = 1, [NCAPI_WES_←](#)
[REQUESTSTATUS](#) = 2, [NCAPI_WES_STARTCLIENT](#) = 3 }
- enum [NcApiWesStatusValues](#) { [NCAPI_WES_STOPPED](#) = 0, [NCAPI_WES_SERVERRUNNING](#) = 1, [NC←](#)
[API_WES_CLIENTRUNNING](#) = 2 }

- enum `NcApiNodeType` { `NCAPI_NODE_TYPE_NC2400` = 1, `NCAPI_NODE_TYPE_NC1000` = 2, `NCAPI_NODE_TYPE_NC0400` = 3 }
- enum `NcApiAltCmdValues` { `NCAPI_ALT_STOP` = 0, `NCAPI_ALT_START` = 1 }
- enum `NcApiNetCmdValues` { `NCAPI_NetCmd_ACK` = 0, `NCAPI_NetCmd_NACK` = 1, `NCAPI_NetCmd_Hibernate` = 2, `NCAPI_NetCmd_Wake` = 3, `NCAPI_NetCmd_Wes` = 5 }

Functions

- void `NcApiSupportMessageReceived` (uint8_t n, void *callbackToken, uint8_t *msg, uint8_t msgLength)
Application provided function that `NcApi` calls after it has successfully received a full message.
- void `NcApiCallbackNwuActive` (uint8_t n)
Callback from the application into `NcApi` whenever nWU becomes active.
- void `NcApiCtsActive` (uint8_t n)
Callback from the application into `NcApi` whenever CTS becomes active.
- void `NcApiSupportTxData` (uint8_t n, uint8_t *finalMsg, uint8_t finalMsgLength)
Application provided function that `NcApi` calls if there is any pending data to be written to the UART.
- void `NcApiSupportMessageWritten` (uint8_t n, void *callbackToken, uint8_t *finalMsg, uint8_t finalMsgLength)
Application provided function that `NcApi` calls after it has successfully written the message.
- void `NcApiRxData` (uint8_t n, uint8_t byte)
Callback from the application into `NcApi` whenever a byte is received on the UART.
- void `NcApiInit` ()
API provided function that initializes the allocated instances of `tNcApi`.
- `NcApiErrorCodes` `NcApiSendUnacknowledged` (uint8_t n, `tNcApiSendUnackParams` *args)
API provided function that may be called when a message type "0x02: Unacknowledged Packet" shall be sent.
- `NcApiErrorCodes` `NcApiSendAcknowledged` (uint8_t n, `tNcApiSendAckParams` *args)
API provided function that may be called when a message type "0x03: Acknowledged Packet" shall be sent.
- `NcApiErrorCodes` `NcApiSendNodeInfoRequest` (uint8_t n, `tNcApiNodeInfoParams` *args)
API provided function that may be called when a message type "0x08: Node Info Request" shall be sent.
- `NcApiErrorCodes` `NcApiSendNeighborListRequest` (uint8_t n, void *callbackToken)
API provided function that may be called when a message type "0x09: Neighbor List Request" shall be sent.
- `NcApiErrorCodes` `NcApiSendNetCmd` (uint8_t n, `tNcApiNetCmdParams` *args)
API provided function that may be called when a message type "0x0a: Network Command" shall be sent.
- `NcApiErrorCodes` `NcApiSendWesCmd` (uint8_t n, `tNcApiWesCmdParams` *args)
API provided function that may be called when a message type "0x10: WES Command" shall be sent.
- `NcApiErrorCodes` `NcApiSendWesResponse` (uint8_t n, `tNcApiWesResponseParams` *args)
API provided function that may be called when a message type "0x11: WES Setup Response" shall be sent.
- `NcApiErrorCodes` `NcApiSendAltCmd` (uint8_t n, `tNcApiAltCmdParams` *args)
Sends one AltCmd message.
- void `NcApiExecuteCallbacks` (uint8_t n, uint8_t *msg, uint8_t msgLength)
API provided function that may be called to the relevant callback functions for a received message.
- void `NcApiCancelEnqueuedMessage` (uint8_t n)
Cancels any enqueued message.
- `NcApiErrorCodes` `NcApiSendRaw` (uint8_t n, `tNcApiSendAckParams` *args)
(This function is not supported)

Variables

- `tNcApi g_ncApi []`
Application defined array of `NcApi` instances in use.
- `uint8_t g_numberOfNcApis`
Application defined number of elements in the `g_ncApi` array.

6.3.1 Typedef Documentation

6.3.1.1 `typedef void(* pfnNcApiHostAckCallback) (uint8_t n, tNcApiHostAckNack *m)`

The appropriate function is called when a HostAck or HostNack message has been received for a previously sent payload package. The callback function delivers a pointer to a struct containing the relevant information.

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
<i>m</i>	Strongly typed message

6.3.1.2 `typedef void(* pfnNcApiHostDataCallback) (uint8_t n, tNcApiHostData *m)`

The callback is issued when the modules receive payload data, that requires acknowledge, from another module in the NEOCORTEC mesh network. The callback function delivers a pointer to a struct containing the relevant information.

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
<i>m</i>	Strongly typed message

6.3.1.3 `typedef void(* pfnNcApiHostDataHapaCallback) (uint8_t n, tNcApiHostDataHapa *m)`

The callback is issued when the modules receive payload data, that requires acknowledge, from another module in the NEOCORTEC mesh network which has been configured to use the High Precision Packet Age feature (HAPA). The callback function delivers a pointer to a struct containing the relevant information.

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
<i>m</i>	Strongly typed message

6.3.1.4 `typedef void(* pfnNcApiHostUappDataCallback) (uint8_t n, tNcApiHostUappData *m)`

The callback is issued when the modules receive payload data, that NOT requires acknowledge, from another module in the NEOCORTEC mesh network. The callback function delivers a pointer to a struct containing the relevant information.

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
<i>m</i>	Strongly typed message

6.3.1.5 typedef void(* pfnNcApiHostUappDataHapaCallback) (uint8_t n, tNcApiHostUappDataHapa *m)

The callback is issued when the modules receive payload data, that requires acknowledge, from another module in the NEOCORTEC mesh network which has been configured to use the High Precision Packet Age feature (HAPA). The callback function delivers a pointer to a struct containing the relevant information.

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
<i>m</i>	Strongly typed message

6.3.1.6 typedef void(* pfnNcApiNeighborListReplyCallback) (uint8_t n, tNcApiNeighborListReply *m)

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
<i>m</i>	Strongly typed message

6.3.1.7 typedef void(* pfnNcApiNetCmdResponseCallback) (uint8_t n, tNcApiNetCmdReply *m)

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
<i>m</i>	Strongly typed message

6.3.1.8 typedef void(* pfnNcApiNodeInfoReplyCallback) (uint8_t n, tNcApiNodeInfoReply *m)

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
<i>m</i>	Strongly typed message

6.3.1.9 typedef void(* pfnNcApiReadCallback) (uint8_t n, uint8_t *msg, uint8_t msgLength)

This function will deliver a byte array containing the received raw UART frame. It is normally not necessary to register for this callback, as there are other callbacks which are specific to the various types of application data.

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
<i>msg</i>	Pointer to the message
<i>msgLength</i>	Message length in bytes

6.3.1.10 typedef void(* pfnNcApiWesSetupRequestCallback) (uint8_t n, tNcApiWesSetupRequest *m)

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
----------	---

<i>m</i>	Strongly typed message
----------	------------------------

6.3.1.11 `typedef void(* pfnNcApiWesStatusCallback) (uint8_t n, tNcApiWesStatus *m)`

Parameters

<i>n</i>	Index of tNcApi instance that the message was received from
<i>m</i>	Strongly typed message

6.3.1.12 `typedef struct NcApi tNcApi`

6.3.1.13 `typedef struct NcApiAltCmdMessage tNcApiAltCmdMessage`

6.3.1.14 `typedef struct NcApiAltCmdParams tNcApiAltCmdParams`

6.3.1.15 `typedef struct NcApiHostAckNack tNcApiHostAckNack`

6.3.1.16 `typedef struct NcApiHostData tNcApiHostData`

6.3.1.17 `typedef struct NcApiHostDataHapa tNcApiHostDataHapa`

6.3.1.18 `typedef struct NcApiHostUappData tNcApiHostUappData`

6.3.1.19 `typedef struct NcApiHostUappDataHapa tNcApiHostUappDataHapa`

6.3.1.20 `typedef struct NcApiNeighbor tNcApiNeighbor`

6.3.1.21 `typedef struct NcApiNeighborListReply tNcApiNeighborListReply`

6.3.1.22 `typedef struct NcApiNeighborListRequestMessage tNcApiNeighborListRequestMessage`

6.3.1.23 `typedef struct NcApiNeighborListRequestParams tNcApiNeighborListRequestParams`

6.3.1.24 `typedef struct NcApiNetCmdMessage tNcApiNetCmdMessage`

6.3.1.25 `typedef struct NcApiNetCmdParams tNcApiNetCmdParams`

6.3.1.26 `typedef struct NcApiNetCmdReply tNcApiNetCmdReply`

6.3.1.27 `typedef struct NcApiNodeInfoParams tNcApiNodeInfoParams`

6.3.1.28 `typedef struct NcApiNodeInfoReply tNcApiNodeInfoReply`

6.3.1.29 `typedef struct NcApiNodeInfoRequestMessage tNcApiNodeInfoRequestMessage`

6.3.1.30 `typedef struct NcApiRxHandlers tNcApiRxHandlers`

6.3.1.31 `typedef struct NcApiSendAckMessage tNcApiSendAckMessage`

6.3.1.32 `typedef struct NcApiSendAckParams tNcApiSendAckParams`

6.3.1.33 `typedef struct NcApiSendUnackMessage tNcApiSendUnackMessage`

6.3.1.34 typedef struct NcApiSendUnackParams tNcApiSendUnackParams

6.3.1.35 typedef struct NcApiWesCmdMessage tNcApiWesCmdMessage

6.3.1.36 typedef struct NcApiWesCmdParams tNcApiWesCmdParams

6.3.1.37 typedef struct NcApiWesResponseMessage tNcApiWesResponseMessage

6.3.1.38 typedef struct NcApiWesResponseParams tNcApiWesResponseParams

6.3.1.39 typedef struct NcApiWesSetupRequest tNcApiWesSetupRequest

6.3.1.40 typedef struct NcApiWesStatus tNcApiWesStatus

6.3.2 Enumeration Type Documentation

6.3.2.1 enum NcApiAltCmdValues

Enumerator

NCAPI_ALT_STOP Stop.

NCAPI_ALT_START Start server.

6.3.2.2 enum NcApiErrorCodes

Enumerator

NCAPI_OK Success.

NCAPI_ERR_NODEID NodeId cannot be 0.

NCAPI_ERR_DESTPORT Port must be [0..3].

NCAPI_ERR_PAYLOAD No payload supplied.

NCAPI_ERR_ENQUEUED There is already one pending message waiting to be written to the UART.

NCAPI_ERR_NULLPAYLOAD Payload length supplied but no payload.

NCAPI_ERR_NOARGS No arguments pointer.

6.3.2.3 enum NcApiNetCmdValues

Enumerator

NCAPI_NetCmd_ACK ACK.

NCAPI_NetCmd_NACK NACK.

NCAPI_NetCmd_Hibernate Hibernate.

NCAPI_NetCmd_Wake Wake.

NCAPI_NetCmd_Wes WES.

6.3.2.4 enum NcApiNodeType

Enumerator

NCAPI_NODE_TYPE_NC2400 Hardware is NC2400.

NCAPI_NODE_TYPE_NC1000 Hardware is NC1000.

NCAPI_NODE_TYPE_NC0400 Hardware is NC0400.

6.3.2.5 enum NcApiWesCmdValues

Enumerator

NCAPI_WES_STOP Stop.
NCAPI_WES_STARTSERVER Start server.
NCAPI_WES_REQUESTSTATUS Request status.
NCAPI_WES_STARTCLIENT Start client.

6.3.2.6 enum NcApiWesStatusValues

Enumerator

NCAPI_WES_STOPPED Stopped.
NCAPI_WES_SERVERRUNNING Server is running.
NCAPI_WES_CLIENTRUNNING Client is running.

6.3.3 Function Documentation

6.3.3.1 void NcApiCallbackNwuActive (uint8_t n)

Parameters

<i>n</i>	Index of tNcApi instance that the nWU interrupt relates to
----------	--

6.3.3.2 void NcApiCancelEnqueuedMessage (uint8_t n)

Parameters

<i>n</i>	Index of tNcApi instance where the message should be dequeued
----------	---

6.3.3.3 void NcApiCtsActive (uint8_t n)

This is a function inside the API, which shall be called each time the CTS line on the UART transitions to active (low) state. As such, a monitor for the particular pin on the controller must be present, such that the API can be made aware that the CTS is active. There is no need for any action when the CTS line transitions to passive (high).

Parameters

<i>n</i>	Index of tNcApi instance that the CTS interrupt relates to
----------	--

6.3.3.4 void NcApiExecuteCallbacks (uint8_t n, uint8_t * msg, uint8_t msgLength)

The Application typically calls this function from [NcApiSupportMessageReceived\(\)](#) once a message has successfully been received from the module, and the relevant callback function in the Application will then be called, if it has previously been registered.

Parameters

<i>n</i>	Index of tNcApi instance that message was written to
<i>msg</i>	Pointer to the message
<i>msgLength</i>	Message length in bytes

6.3.3.5 void NcApiInit ()

Remarks

This function must be called once to initialize the API.

6.3.3.6 void NcApiRxData (uint8_t n, uint8_t byte)

This is a function inside the API, which shall be called each time a byte is received on the UART. The byte shall be included as an argument when the function is called.

Parameters

<i>n</i>	Index of tNcApi instance that the byte relates to
<i>byte</i>	The byte received

6.3.3.7 NcApiErrorCodes NcApiSendAcknowledged (uint8_t n, tNcApiSendAckParams * args)

This function can be used to send payload data to another node inside the mesh network. The message will be acknowledged.

Parameters

<i>n</i>	Index of tNcApi instance that the message should be sent via
<i>args</i>	Pointer to instance of the argument structure that holds the parameters

Returns

0 upon success. Anything else is an error

6.3.3.8 NcApiErrorCodes NcApiSendAltCmd (uint8_t n, tNcApiAltCmdParams * args)

Parameters

<i>n</i>	Index of tNcApi instance that the message should be sent via
<i>args</i>	Pointer to instance of the argument structure that holds the parameters

Returns

0 upon success. Anything else is an error

6.3.3.9 NcApiErrorCodes NcApiSendNeighborListRequest (uint8_t n, void * callbackToken)

Parameters

<i>n</i>	Index of tNcApi instance that the message should be sent via
<i>args</i>	Pointer to instance of the argument structure that holds the parameters

Returns

0 upon success. Anything else is an error

6.3.3.10 **NcApiErrorCodes** NcApiSendNetCmd (uint8_t *n*, tNcApiNetCmdParams * *args*)

Parameters

<i>n</i>	Index of tNcApi instance that the message should be sent via
<i>args</i>	Pointer to instance of the argument structure that holds the parameters

Returns

0 upon success. Anything else is an error

6.3.3.11 **NcApiErrorCodes** NcApiSendNodeInfoRequest (uint8_t *n*, tNcApiNodeInfoParams * *args*)

Parameters

<i>n</i>	Index of tNcApi instance that the message should be sent via
<i>args</i>	Pointer to instance of the argument structure that holds the parameters

Returns

0 upon success. Anything else is an error

6.3.3.12 **NcApiErrorCodes** NcApiSendRaw (uint8_t *n*, tNcApiSendAckParams * *args*)

6.3.3.13 **NcApiErrorCodes** NcApiSendUnacknowledged (uint8_t *n*, tNcApiSendUnackParams * *args*)

This function can be used to send payload data to another node inside the mesh network. The message will not be acknowledged.

Parameters

<i>n</i>	Index of tNcApi instance that the message should be sent via
<i>args</i>	Pointer to instance of the argument structure that holds the parameters

Returns

0 upon success. Anything else is an error

6.3.3.14 **NcApiErrorCodes** NcApiSendWesCmd (uint8_t *n*, tNcApiWesCmdParams * *args*)

Parameters

<i>n</i>	Index of tNcApi instance that the message should be sent via
<i>args</i>	Pointer to instance of the argument structure that holds the parameters

<i>n</i>	Index of tNcApi instance that the message should be sent via
<i>args</i>	Pointer to instance of the argument structure that holds the parameters

Returns

0 upon success. Anything else is an error

6.3.3.15 NcApiErrorCodes NcApiSendWesResponse (uint8_t *n*, tNcApiWesResponseParams * *args*)

Parameters

<i>n</i>	Index of tNcApi instance that the message should be sent via
<i>args</i>	Pointer to instance of the argument structure that holds the parameters

Returns

0 upon success. Anything else is an error

6.3.3.16 void NcApiSupportMessageReceived (uint8_t *n*, void * *callbackToken*, uint8_t * *msg*, uint8_t *msgLength*)

The API will call this function once a message has successfully been received in full from the module. This can be used by the application layer to initiate invocation of the relevant callback function via the function [NcApiExecute↔Callbacks\(\)](#).

Parameters

<i>n</i>	Index of tNcApi instance that message was written to
<i>callbackToken</i>	Application provided context / token / tag
<i>msg</i>	Pointer to the message
<i>msgLength</i>	Message length in bytes

6.3.3.17 void NcApiSupportMessageWritten (uint8_t *n*, void * *callbackToken*, uint8_t * *finalMsg*, uint8_t *finalMsgLength*)

The API will call this function once a message has successfully been written in full to the module. This can be used by the application layer to check that a previous request to send a message was completed successfully.

Parameters

<i>n</i>	Index of tNcApi instance that message was written to
<i>callbackToken</i>	Application provided context / token / tag
<i>finalMsg</i>	Pointer to the message
<i>finalMsgLength</i>	Message length in bytes

6.3.3.18 void NcApiSupportTxData (uint8_t *n*, uint8_t * *finalMsg*, uint8_t *finalMsgLength*)

The API will call this function to send data to the UART. The function is called with a pointer to the actual data to be written. The function shall implement the necessary code required to output the data to the UART.

Parameters

<i>n</i>	Index of tNcApi instance that the data should be written to, ie. which UART
----------	---

<i>finalMsg</i>	Pointer to the buffer
<i>finalMsgLength</i>	Number of bytes to be written

6.3.4 Variable Documentation

6.3.4.1 `tNcApi g_ncApi[]`

6.3.4.2 `uint8_t g_numberOfNcApis`

6.4 NeoParser.h File Reference

```
#include <stdint.h>
#include "NcApi.h"
```

Macros

- `#define NCAPI_HOST_PREFIX_SIZE 2`
- `#define NCAPI_HOSTSTACK_LENGTH 2`
- `#define NCAPI_HOSTDATA_HEADER_SIZE 5`
- `#define NCAPI_HOSTDATA_MIN_LENGTH (NCAPI_HOSTDATA_HEADER_SIZE+NCAPI_HOST_PREFIX_SIZE)`
- `#define NCAPI_HOSTDATAHAPA_HEADER_SIZE 7`
- `#define NCAPI_HOSTDATAHAPA_MIN_LENGTH (NCAPI_HOSTDATAHAPA_HEADER_SIZE+NCAPI_HOST_PREFIX_SIZE)`
- `#define NCAPI_HOSTUAPPDATA_HEADER_SIZE 7`
- `#define NCAPI_HOSTUAPPDATA_MIN_LENGTH (NCAPI_HOSTUAPPDATA_HEADER_SIZE+NCAPI_HOST_PREFIX_SIZE)`
- `#define NCAPI_HOSTUAPPDATAHAPA_HEADER_SIZE 9`
- `#define NCAPI_HOSTUAPPDATAHAPA_MIN_LENGTH (NCAPI_HOSTUAPPDATAHAPA_HEADER_SIZE+NCAPI_HOST_PREFIX_SIZE)`
- `#define NCAPI_NODEINFOREQUEST_LENGTH 0`
- `#define NCAPI_NEIGHBORLISTREQUEST_LENGTH 0`
- `#define NCAPI_NETCMD_LENGTH unused`
- `#define NCAPI_WESCMD_LENGTH 1`
- `#define NCAPI_WESSETUPREQUEST_LENGTH 6`
- `#define NCAPI_ALTCMD_LENGTH 1`
- `#define NCAPI_NODEINFOREPLY_LENGTH 8`
- `#define NCAPI_WESRESPONSE_LENGTH (7 + WES_APPSETTINGS_LENGTH)`
- `#define NCAPI_WESSTATUS_LENGTH 1`
- `#define NCAPI_NEIGHBORLISTREPLY_LENGTH 36`
- `#define NCAPI_NEIGHBORLISTREPLY_EX_LENGTH 39`
- `#define NCAPI_NETCMDRESPONSE_HEADER_SIZE 3`
- `#define NCAPI_NETCMDRESPONSE_MIN_LENGTH 5`

Typedefs

- `typedef enum NcApiMessageType NcApiMessageType`

Enumerations

- enum [NcApiMessageType](#) {
[CommandUnacknowledgedEnum](#) = 0x02, [CommandAcknowledgedEnum](#) = 0x03, [NodeInfoRequestEnum](#) = 0x08, [NeighborListRequestEnum](#) = 0x09,
[NetCmdEnum](#) = 0x0a, [WesCmdEnum](#) = 0x10, [WesResponseEnum](#) = 0x11, [AltCmdEnum](#) = 0x20,
[HostAckEnum](#) = 0x50, [HostNAckEnum](#) = 0x51, [HostDataEnum](#) = 0x52, [HostDataHapaEnum](#) = 0x53,
[HostUappDataEnum](#) = 0x54, [HostUappDataHapaEnum](#) = 0x55, [NodeInfoReplyEnum](#) = 0x58, [NeighborListReplyEnum](#) = 0x59,
[NetCmdReplyEnum](#) = 0x5a, [WesStatusEnum](#) = 0x60, [WesSetupRequestEnum](#) = 0x61, [CommandRawEnum](#) = 0xff }

Functions

- int [NcApilsValidFrameTraceSpecific](#) (uint8_t *buffer, uint16_t bufLength)
Determines if the buffer contains a valid TraceFrame message.
- int [NcApilsValidApiFrame](#) (uint8_t *buffer, uint16_t bufLength, uint16_t *outStartAt, uint16_t *outLength)
Determines if the content in the buffer is a valid Protocol-message.
- int [NcApilsValidSysFrame](#) (uint8_t *buffer, uint16_t bufLength, uint16_t *outStartAt, uint16_t *outLength)
Determines if the content in the buffer is a valid System-trace-message or a Bootloader-message.
- int [NcApilsMsgReadOtpReply](#) (uint8_t *buffer, uint16_t startAt, uint16_t length)
Determines whether the message in the buffer contains Uid.
- int [NcApilsMsgGetBootLoaderVersionReply](#) (uint8_t *buffer, uint16_t startAt, uint16_t length)
Determines whether the message in the buffer contains bootloaderversion.
- int [NcApilsMsgGetProtocolVersionReply](#) (uint8_t *buffer, uint16_t startAt, uint16_t length)
Determines whether the message in the buffer contains protocolversion.
- void [NcApiGetMsgAsHostAck](#) (uint8_t *buffer, tNcApiHostAckNack *p)
Deserializes the content of the buffer into a corresponding structure.
- void [NcApiGetMsgAsHostData](#) (uint8_t *buffer, tNcApiHostData *p)
Deserializes the content of the buffer into a corresponding structure.
- void [NcApiGetMsgAsHostDataHapa](#) (uint8_t *buffer, tNcApiHostDataHapa *p)
Deserializes the content of the buffer into a corresponding structure.
- void [NcApiGetMsgAsHostUappData](#) (uint8_t *buffer, tNcApiHostUappData *p)
Deserializes the content of the buffer into a corresponding structure.
- void [NcApiGetMsgAsHostUappDataHapa](#) (uint8_t *buffer, tNcApiHostUappDataHapa *p)
Deserializes the content of the buffer into a corresponding structure.
- void [NcApiGetMsgAsNodeInfoReply](#) (uint8_t *buffer, tNcApiNodeInfoReply *p)
Deserializes the content of the buffer into a corresponding structure.
- void [NcApiGetMsgAsWesStatus](#) (uint8_t *buffer, tNcApiWesStatus *p)
Deserializes the content of the buffer into a corresponding structure.
- void [NcApiGetMsgAsWesSetupRequest](#) (uint8_t *buffer, tNcApiWesSetupRequest *p)
Deserializes the content of the buffer into a corresponding structure.
- void [NcApiGetMsgAsNodeInfo](#) (uint8_t *buffer, tNcApiNodeInfoReply *p)
Deserializes the content of the buffer into a corresponding structure.
- void [NcApiGetMsgAsNeighborListReply](#) (uint8_t *buffer, tNcApiNeighborListReply *p)
Deserializes the content of the buffer into a corresponding structure.
- void [NcApiGetMsgAsNetCmdResponse](#) (uint8_t *buffer, tNcApiNetCmdReply *p)
Deserializes the content of the buffer into a corresponding structure.

6.4.1 Typedef Documentation

6.4.1.1 typedef enum NcApiMessageType NcApiMessageType

6.4.2 Enumeration Type Documentation

6.4.2.1 enum NcApiMessageType

Enumerator

CommandUnacknowledgedEnum

CommandAcknowledgedEnum

NodeInfoRequestEnum

NeighborListRequestEnum

NetCmdEnum

WesCmdEnum

WesResponseEnum

AltCmdEnum

HostAckEnum

HostNAckEnum

HostDataEnum

HostDataHapaEnum

HostUappDataEnum

HostUappDataHapaEnum

NodeInfoReplyEnum

NeighborListReplyEnum

NetCmdReplyEnum

WesStatusEnum

WesSetupRequestEnum

CommandRawEnum

6.4.3 Function Documentation

6.4.3.1 void NcApiGetMsgAsHostAck (uint8_t * *buffer*, tNcApiHostAckNack * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.2 void NcApiGetMsgAsHostData (uint8_t * *buffer*, tNcApiHostData * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.3 void NcApiGetMsgAsHostDataHapa (uint8_t * *buffer*, tNcApiHostDataHapa * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.4 void NcApiGetMsgAsHostUappData (uint8_t * *buffer*, tNcApiHostUappData * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.5 void NcApiGetMsgAsHostUappDataHapa (uint8_t * *buffer*, tNcApiHostUappDataHapa * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.6 void NcApiGetMsgAsNeighborListReply (uint8_t * *buffer*, tNcApiNeighborListReply * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.7 void NcApiGetMsgAsNetCmdResponse (uint8_t * *buffer*, tNcApiNetCmdReply * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.8 void NcApiGetMsgAsNodeInfo (uint8_t * *buffer*, tNcApiNodeInfoReply * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.9 void NcApiGetMsgAsNodeInfoReply (uint8_t * *buffer*, tNcApiNodeInfoReply * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.10 void NcApiGetMsgAsWesSetupRequest (uint8_t * *buffer*, tNcApiWesSetupRequest * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.11 void NcApiGetMsgAsWesStatus (uint8_t * *buffer*, tNcApiWesStatus * *p*)

Parameters

<i>buffer</i>	Buffer containing message
<i>p</i>	Instance to deserialize into

6.4.3.12 int NcApilsMsgGetBootLoaderVersionReply (uint8_t * *buffer*, uint16_t *startAt*, uint16_t *length*)

Parameters

<i>buffer</i>	Buffer containing message
<i>startAt</i>	Index of message start in buffer
<i>length</i>	Message length

Returns

1==true 0==false

6.4.3.13 int NcApilsMsgGetProtocolVersionReply (uint8_t * *buffer*, uint16_t *startAt*, uint16_t *length*)

Parameters

<i>buffer</i>	Buffer containing message
<i>startAt</i>	Index of message start in buffer
<i>length</i>	Message length

Returns

1==true 0==false

6.4.3.14 int NcApilsMsgReadOtpReply (uint8_t * *buffer*, uint16_t *startAt*, uint16_t *length*)

Parameters

<i>buffer</i>	Buffer containing message
<i>startAt</i>	Index of message start in buffer
<i>length</i>	Message length

Returns

1==true 0==false

6.4.3.15 int NcApilsValidApiFrame (uint8_t * *buffer*, uint16_t *bufLength*, uint16_t * *outStartAt*, uint16_t * *outLength*)

Parameters

	<i>buffer</i>	Received RX-data
	<i>bufLength</i>	Number of received bytes
out	<i>outStartAt</i>	If a message was found, index into buffer where the message begins
out	<i>outLength</i>	If a message was found, the message length

Returns

1==true 0==false

6.4.3.16 int NcApilsValidFrameTraceSpecific (uint8_t * *buffer*, uint16_t *bufLength*)

Parameters

<i>buffer</i>	Received RX-data
<i>bufLength</i>	Buffer length

Returns

1==true 0==false

6.4.3.17 int NcApilsValidSysFrame (uint8_t * *buffer*, uint16_t *bufLength*, uint16_t * *outStartAt*, uint16_t * *outLength*)

Parameters

	<i>buffer</i>	Received RX-data
	<i>bufLength</i>	Number of received bytes
out	<i>outStartAt</i>	If a message was found, index into buffer where the message begins
out	<i>outLength</i>	If a message was found, the message length

Returns

1==true 0==false

Index

AltCmdEnum
 NeoParser.h, [40](#)

CommandAcknowledgedEnum
 NeoParser.h, [40](#)

CommandRawEnum
 NeoParser.h, [40](#)

CommandUnacknowledgedEnum
 NeoParser.h, [40](#)

DoxygenMainPage.h, [25](#)

DoxygenPage001.h, [25](#)

g_ncApi
 NcApi.h, [38](#)

g_numberOfNcApis
 NcApi.h, [38](#)

HostAckEnum
 NeoParser.h, [40](#)

HostDataEnum
 NeoParser.h, [40](#)

HostDataHapaEnum
 NeoParser.h, [40](#)

HostNAckEnum
 NeoParser.h, [40](#)

HostUappDataEnum
 NeoParser.h, [40](#)

HostUappDataHapaEnum
 NeoParser.h, [40](#)

NCAPI_ALT_START
 NcApi.h, [33](#)

NCAPI_ALT_STOP
 NcApi.h, [33](#)

NCAPI_ERR_DESTPORT
 NcApi.h, [33](#)

NCAPI_ERR_ENQUEUED
 NcApi.h, [33](#)

NCAPI_ERR_NOARGS
 NcApi.h, [33](#)

NCAPI_ERR_NODEID
 NcApi.h, [33](#)

NCAPI_ERR_NULLPAYLOAD
 NcApi.h, [33](#)

NCAPI_ERR_PAYLOAD
 NcApi.h, [33](#)

NCAPI_NODE_TYPE_NC0400
 NcApi.h, [33](#)

NCAPI_NODE_TYPE_NC1000
 NcApi.h, [33](#)

NCAPI_NODE_TYPE_NC2400
 NcApi.h, [33](#)

NCAPI_NetCmd_ACK
 NcApi.h, [33](#)

NCAPI_NetCmd_Hibernate
 NcApi.h, [33](#)

NCAPI_NetCmd_NACK
 NcApi.h, [33](#)

NCAPI_NetCmd_Wake
 NcApi.h, [33](#)

NCAPI_NetCmd_Wes
 NcApi.h, [33](#)

NCAPI_OK
 NcApi.h, [33](#)

NCAPI_WES_CLIENTRUNNING
 NcApi.h, [34](#)

NCAPI_WES_REQUESTSTATUS
 NcApi.h, [34](#)

NCAPI_WES_SERVERRUNNING
 NcApi.h, [34](#)

NCAPI_WES_STARTCLIENT
 NcApi.h, [34](#)

NCAPI_WES_STARTSERVER
 NcApi.h, [34](#)

NCAPI_WES_STOP
 NcApi.h, [34](#)

NCAPI_WES_STOPPED
 NcApi.h, [34](#)

NcApi, [13](#)

NcApi.h, [25](#)

 g_ncApi, [38](#)

 g_numberOfNcApis, [38](#)

 NCAPI_ALT_START, [33](#)

 NCAPI_ALT_STOP, [33](#)

 NCAPI_ERR_DESTPORT, [33](#)

 NCAPI_ERR_ENQUEUED, [33](#)

 NCAPI_ERR_NOARGS, [33](#)

 NCAPI_ERR_NODEID, [33](#)

 NCAPI_ERR_NULLPAYLOAD, [33](#)

 NCAPI_ERR_PAYLOAD, [33](#)

 NCAPI_NODE_TYPE_NC0400, [33](#)

 NCAPI_NODE_TYPE_NC1000, [33](#)

 NCAPI_NODE_TYPE_NC2400, [33](#)

 NCAPI_NetCmd_ACK, [33](#)

 NCAPI_NetCmd_Hibernate, [33](#)

 NCAPI_NetCmd_NACK, [33](#)

 NCAPI_NetCmd_Wake, [33](#)

 NCAPI_NetCmd_Wes, [33](#)

 NCAPI_OK, [33](#)

NCAPI_WES_CLIENTRUNNING, 34
 NCAPI_WES_REQUESTSTATUS, 34
 NCAPI_WES_SERVERRUNNING, 34
 NCAPI_WES_STARTCLIENT, 34
 NCAPI_WES_STARTSERVER, 34
 NCAPI_WES_STOP, 34
 NCAPI_WES_STOPPED, 34
 NcApiAltCmdValues, 33
 NcApiCallbackNwuActive, 34
 NcApiCancelEnqueuedMessage, 34
 NcApiCtsActive, 34
 NcApiErrorCodes, 33
 NcApiExecuteCallbacks, 34
 NcApiInit, 35
 NcApiNetCmdValues, 33
 NcApiNodeType, 33
 NcApiRxData, 35
 NcApiSendAcknowledged, 35
 NcApiSendAltCmd, 35
 NcApiSendNeighborListRequest, 35
 NcApiSendNetCmd, 36
 NcApiSendNodeInfoRequest, 36
 NcApiSendRaw, 36
 NcApiSendUnacknowledged, 36
 NcApiSendWesCmd, 36
 NcApiSendWesResponse, 37
 NcApiSupportMessageReceived, 37
 NcApiSupportMessageWritten, 37
 NcApiSupportTxData, 37
 NcApiWesCmdValues, 33
 NcApiWesStatusValues, 34
 pfnNcApiHostAckCallback, 30
 pfnNcApiHostDataCallback, 30
 pfnNcApiHostDataHapaCallback, 30
 pfnNcApiHostUappDataCallback, 30
 pfnNcApiHostUappDataHapaCallback, 30
 pfnNcApiNeighborListReplyCallback, 31
 pfnNcApiNetCmdResponseCallback, 31
 pfnNcApiNodeInfoReplyCallback, 31
 pfnNcApiReadCallback, 31
 pfnNcApiWesSetupRequestCallback, 31
 pfnNcApiWesStatusCallback, 32
 tNcApi, 32
 tNcApiAltCmdMessage, 32
 tNcApiAltCmdParams, 32
 tNcApiHostAckNack, 32
 tNcApiHostData, 32
 tNcApiHostDataHapa, 32
 tNcApiHostUappData, 32
 tNcApiHostUappDataHapa, 32
 tNcApiNeighbor, 32
 tNcApiNeighborListReply, 32
 tNcApiNeighborListRequestMessage, 32
 tNcApiNeighborListRequestParams, 32
 tNcApiNetCmdMessage, 32
 tNcApiNetCmdParams, 32
 tNcApiNetCmdReply, 32
 tNcApiNodeInfoParams, 32
 tNcApiNodeInfoReply, 32
 tNcApiNodeInfoRequestMessage, 32
 tNcApiRxHandlers, 32
 tNcApiSendAckMessage, 32
 tNcApiSendAckParams, 32
 tNcApiSendUnackMessage, 32
 tNcApiSendUnackParams, 32
 tNcApiWesCmdMessage, 33
 tNcApiWesCmdParams, 33
 tNcApiWesResponseMessage, 33
 tNcApiWesResponseParams, 33
 tNcApiWesSetupRequest, 33
 tNcApiWesStatus, 33
 NcApiAltCmdMessage, 13
 NcApiAltCmdParams, 14
 NcApiAltCmdValues
 NcApi.h, 33
 NcApiCallbackNwuActive
 NcApi.h, 34
 NcApiCancelEnqueuedMessage
 NcApi.h, 34
 NcApiCtsActive
 NcApi.h, 34
 NcApiErrorCodes
 NcApi.h, 33
 NcApiExecuteCallbacks
 NcApi.h, 34
 NcApiGetMsgAsHostAck
 NeoParser.h, 40
 NcApiGetMsgAsHostData
 NeoParser.h, 40
 NcApiGetMsgAsHostDataHapa
 NeoParser.h, 40
 NcApiGetMsgAsHostUappData
 NeoParser.h, 41
 NcApiGetMsgAsHostUappDataHapa
 NeoParser.h, 41
 NcApiGetMsgAsNeighborListReply
 NeoParser.h, 41
 NcApiGetMsgAsNetCmdResponse
 NeoParser.h, 41
 NcApiGetMsgAsNodeInfo
 NeoParser.h, 41
 NcApiGetMsgAsNodeInfoReply
 NeoParser.h, 41
 NcApiGetMsgAsWesSetupRequest
 NeoParser.h, 41
 NcApiGetMsgAsWesStatus
 NeoParser.h, 42
 NcApiHostAckNack, 14
 NcApiHostData, 14
 NcApiHostDataHapa, 15
 NcApiHostUappData, 15
 NcApiHostUappDataHapa, 15
 NcApiInit
 NcApi.h, 35
 NcApiIlsMsgGetBootLoaderVersionReply
 NeoParser.h, 42

- NcApiMsgGetProtocolVersionReply
 - NeoParser.h, [42](#)
- NcApiMsgReadOtpReply
 - NeoParser.h, [42](#)
- NcApiValidApiFrame
 - NeoParser.h, [42](#)
- NcApiValidFrameTraceSpecific
 - NeoParser.h, [43](#)
- NcApiValidSysFrame
 - NeoParser.h, [43](#)
- NcApiMessageType
 - NeoParser.h, [40](#)
- NcApiNeighbor, [16](#)
- NcApiNeighborListReply, [16](#)
- NcApiNeighborListRequestMessage, [16](#)
- NcApiNeighborListRequestParams, [17](#)
- NcApiNetCmdMessage, [17](#)
- NcApiNetCmdParams, [17](#)
- NcApiNetCmdReply, [18](#)
- NcApiNetCmdValues
 - NcApi.h, [33](#)
- NcApiNodeInfoParams, [18](#)
- NcApiNodeInfoReply, [19](#)
- NcApiNodeInfoRequestMessage, [19](#)
- NcApiNodeType
 - NcApi.h, [33](#)
- NcApiRxData
 - NcApi.h, [35](#)
- NcApiRxHandlers, [19](#)
- NcApiSendAckMessage, [20](#)
- NcApiSendAckParams, [20](#)
- NcApiSendAcknowledged
 - NcApi.h, [35](#)
- NcApiSendAltCmd
 - NcApi.h, [35](#)
- NcApiSendNeighborListRequest
 - NcApi.h, [35](#)
- NcApiSendNetCmd
 - NcApi.h, [36](#)
- NcApiSendNodeInfoRequest
 - NcApi.h, [36](#)
- NcApiSendRaw
 - NcApi.h, [36](#)
- NcApiSendUnackMessage, [21](#)
- NcApiSendUnackParams, [21](#)
- NcApiSendUnacknowledged
 - NcApi.h, [36](#)
- NcApiSendWesCmd
 - NcApi.h, [36](#)
- NcApiSendWesResponse
 - NcApi.h, [37](#)
- NcApiSupportMessageReceived
 - NcApi.h, [37](#)
- NcApiSupportMessageWritten
 - NcApi.h, [37](#)
- NcApiSupportTxData
 - NcApi.h, [37](#)
- NcApiWesCmdMessage, [22](#)
- NcApiWesCmdParams, [22](#)
- NcApiWesCmdValues
 - NcApi.h, [33](#)
- NcApiWesResponseMessage, [22](#)
- NcApiWesResponseParams, [23](#)
- NcApiWesSetupRequest, [23](#)
- NcApiWesStatus, [23](#)
- NcApiWesStatusValues
 - NcApi.h, [34](#)
- NeighborListReplyEnum
 - NeoParser.h, [40](#)
- NeighborListRequestEnum
 - NeoParser.h, [40](#)
- NeoParser.h, [38](#)
 - AltCmdEnum, [40](#)
 - CommandAcknowledgedEnum, [40](#)
 - CommandRawEnum, [40](#)
 - CommandUnacknowledgedEnum, [40](#)
 - HostAckEnum, [40](#)
 - HostDataEnum, [40](#)
 - HostDataHapaEnum, [40](#)
 - HostNackEnum, [40](#)
 - HostUappDataEnum, [40](#)
 - HostUappDataHapaEnum, [40](#)
 - NcApiGetMsgAsHostAck, [40](#)
 - NcApiGetMsgAsHostData, [40](#)
 - NcApiGetMsgAsHostDataHapa, [40](#)
 - NcApiGetMsgAsHostUappData, [41](#)
 - NcApiGetMsgAsHostUappDataHapa, [41](#)
 - NcApiGetMsgAsNeighborListReply, [41](#)
 - NcApiGetMsgAsNetCmdResponse, [41](#)
 - NcApiGetMsgAsNodeInfo, [41](#)
 - NcApiGetMsgAsNodeInfoReply, [41](#)
 - NcApiGetMsgAsWesSetupRequest, [41](#)
 - NcApiGetMsgAsWesStatus, [42](#)
 - NcApiMsgGetBootLoaderVersionReply, [42](#)
 - NcApiMsgGetProtocolVersionReply, [42](#)
 - NcApiMsgReadOtpReply, [42](#)
 - NcApiValidApiFrame, [42](#)
 - NcApiValidFrameTraceSpecific, [43](#)
 - NcApiValidSysFrame, [43](#)
 - NcApiMessageType, [40](#)
 - NeighborListReplyEnum, [40](#)
 - NeighborListRequestEnum, [40](#)
 - NetCmdEnum, [40](#)
 - NetCmdReplyEnum, [40](#)
 - NodeInfoReplyEnum, [40](#)
 - NodeInfoRequestEnum, [40](#)
 - WesCmdEnum, [40](#)
 - WesResponseEnum, [40](#)
 - WesSetupRequestEnum, [40](#)
 - WesStatusEnum, [40](#)
- NetCmdEnum
 - NeoParser.h, [40](#)
- NetCmdReplyEnum
 - NeoParser.h, [40](#)
- NodeInfoReplyEnum
 - NeoParser.h, [40](#)

NodeInfoRequestEnum
 NeoParser.h, 40

pfnNcApiHostAckCallback
 NcApi.h, 30

pfnNcApiHostDataCallback
 NcApi.h, 30

pfnNcApiHostDataHapaCallback
 NcApi.h, 30

pfnNcApiHostUappDataCallback
 NcApi.h, 30

pfnNcApiHostUappDataHapaCallback
 NcApi.h, 30

pfnNcApiNeighborListReplyCallback
 NcApi.h, 31

pfnNcApiNetCmdResponseCallback
 NcApi.h, 31

pfnNcApiNodeInfoReplyCallback
 NcApi.h, 31

pfnNcApiReadCallback
 NcApi.h, 31

pfnNcApiWesSetupRequestCallback
 NcApi.h, 31

pfnNcApiWesStatusCallback
 NcApi.h, 32

tNcApi
 NcApi.h, 32

tNcApiAltCmdMessage
 NcApi.h, 32

tNcApiAltCmdParams
 NcApi.h, 32

tNcApiHostAckNack
 NcApi.h, 32

tNcApiHostData
 NcApi.h, 32

tNcApiHostDataHapa
 NcApi.h, 32

tNcApiHostUappData
 NcApi.h, 32

tNcApiHostUappDataHapa
 NcApi.h, 32

tNcApiNeighbor
 NcApi.h, 32

tNcApiNeighborListReply
 NcApi.h, 32

tNcApiNeighborListRequestMessage
 NcApi.h, 32

tNcApiNeighborListRequestParams
 NcApi.h, 32

tNcApiNetCmdMessage
 NcApi.h, 32

tNcApiNetCmdParams
 NcApi.h, 32

tNcApiNetCmdReply
 NcApi.h, 32

tNcApiNodeInfoParams
 NcApi.h, 32

tNcApiNodeInfoReply
 NcApi.h, 32

tNcApiNodeInfoRequestMessage
 NcApi.h, 32

tNcApiRxHandlers
 NcApi.h, 32

tNcApiSendAckMessage
 NcApi.h, 32

tNcApiSendAckParams
 NcApi.h, 32

tNcApiSendUnackMessage
 NcApi.h, 32

tNcApiSendUnackParams
 NcApi.h, 32

tNcApiWesCmdMessage
 NcApi.h, 33

tNcApiWesCmdParams
 NcApi.h, 33

tNcApiWesResponseMessage
 NcApi.h, 33

tNcApiWesResponseParams
 NcApi.h, 33

tNcApiWesSetupRequest
 NcApi.h, 33

tNcApiWesStatus
 NcApi.h, 33

WesCmdEnum
 NeoParser.h, 40

WesResponseEnum
 NeoParser.h, 40

WesSetupRequestEnum
 NeoParser.h, 40

WesStatusEnum
 NeoParser.h, 40