UG100: EZSP Reference Guide

The EmberZNet Serial Protocol (EZSP) defined in this document is the protocol used by a host application processor to interact with the EmberZNet PRO stack running on a Network Co-Processor (NCP). EZSP messages are sent between the host and the NCP over either a Serial Peripheral Interface (SPI) or a Universal Asynchronous Receiver/Transmitter (UART) interface.

This document is up to date with EmberZNet PRO Release 7.4.2. See section 1 What's New for a list of what has changed since the previous release.

KEY POINTS

- Itemizes what's new for EZSP since the previous release of EmberZNet PRO.
- Defines the fields in an EZSP frame.
- Defines the protocol format, including type definitions, structure definitions, and named values.
- Provides details for all types of EZSP frames: name, ID, description, command parameters, and response parameters.

Contents

1 What's New

The following sections summarize additions, changes, and deletions made to EZSP from EmberZNet PRO Release 7.3.0 to Release 7.4.0.

The difference between the 7.4.0 and 7.4.2 version of this document is a change in Section 6, described below in Changes.

1.1 Additions

Section 3.3 Named Values, EzspValueId

- Added EZSP_VALUE_DELAYED_JOIN_ACTIVATION

Section 16 Green Power Frames

- Added gpSinkTableGetNumberOfActiveEntries

Section 17 Token Interface Frames

- Added gpSecurityTestVectors
- Added tokenFactoryReset

Corresponding additions were made to Section 18, Alphabetical List of Frames, and Section 19, Numeric List of Frames.

1.2 Changes

Section 2.1 EZSP Protocol Version, changed the value of the EZSP_PROTOCOL_VERSION from 12 to 13.

Section 3.1, Type Definitions, changed the alias of sl_zb_sec_man_derived_key_type_t from uint_8 to uint_16.

Section 3.3, Named Values:

- EzspConfigId, changed the description for EZSP_CONFIG_PACKET_BUFFER_COUNt
- EzspValueId, changed the description for EZSP_VALUE_TRANSIENT_KEY_TIMEOUT

Section 4, Configuration Frames, version: Added Zigbee Stack type value (2).

Section 6, Networking Frames, getRadioParameters Command and Response Parameters corrected.

Due to the deletion of Section 17, Secure EZSP Frames, the subsequent sections were renumbered.

1.3 Deletions

Section 3.1, Type Definitions:

- SecureEZSPSecurityType
- SecureEzspSecurityLevel

Section 3.2, Structure Definitions:

- SecureEzspRandomNumber
- SecureEzspSessionId

Section 3.3, Named Values, EzspConfigID, deleted EZSP_CONFIG_END_DEVICE_BIND_TIMEOUT

Section 3.3, Named Values, sl_zb_sc_man_key_type, deleted SL_ZB_MAN_KEY_TYP_SECURE_EZSP_KEY

Section 9 Security Frames

- Deleted following frames
  - getKey
  - getKeyTableEntry

- setKeyTableEntry
- addOrUpdateKeyTableEntry
- addTransientLinkKey
- getTransientLinkKey
- getTransientKeyTableEntry

Section 17 Secure Ezsp Frames (section deleted)

- Deleted following frames
  - setSecurityKey
  - setSecurityParameters
  - resetToFactoryDefaults
  - getSecurityKeyStatus

Corresponding deletions were made in Section 18, Alphabetical List of Frames, and Section 19, Numeric List of Frames.

2 EmberZNet Serial Protocol

All EZSP frames begin with the same three fields: sequence, frame control, and frame ID. The format of the rest of the frame depends on the frame ID. Figure 1 defines the format for all frame IDs. Most of the frames have a fixed length. A few, such as those containing application messages, are of variable length. The frame control indicates the direction of the message (command or response). For commands, the frame control also contains power management information (SPI interface only). For responses, the frame control also contains status information.

The host initiates a two-message transaction by sending a command message to the NCP. The NCP then sends a response message to the host. When connected using the SPI interface, if the NCP needs to communicate a callback to the host, it will indicate this using the interrupt line and then wait for the host to send the `callback` command. When connected using the UART interface, the NCP can send callbacks to the host asynchronously as soon as they occur.

When a command contains an application message, the host must supply a one-byte tag. This tag is used in future commands and responses to refer to the message. For example, when sending a message, the host provides both the message contents and a tag. The tag is then used to report the fate of the message in a later response from the NCP.

Silicon Labs designed EZSP to be very familiar to customers who have used the EmberZNet PRO stack Application Programming Interface (AP)I. The majority of the commands and responses are functionally identical to those found in EmberZNet PRO. The variations are due mainly to the timing differences of running the application on a separate processor across a serial interface.

2.1 EZSP Protocol Version

The EZSP Protocol Version identifies the version number of the EZSP API. This version number changes across EmberZNet PRO software releases when the EZSP API changes in a way that is not backward-compatible. To interoperate, the host and NCP must use compatible EZSP protocol versions. Following NCP reset, the host first issues the `version` command to the NCP to confirm that the two are operating with compatible versions. If they are not, operation cannot proceed. This document describes current EZSP version that is identified by the macro EZSP_PROTOCOL_VERSION and stack type 2 (mesh).

The macro EZSP_PROTOCOL_VERSION is updated to correspond to a change that affects the protocol. The EZSP Protocol Version for this EmberZNet PRO software release is **13**.

2.2 Byte Order

All multiple octet fields are transmitted and received with the least significant octet first, also referred to as “little endian”. This is the same byte order convention specified by 802.15.4 and ZigBee. Note that EUI64 fields are treated as a 64-bit number and are therefore transmitted and received in little endian order. Each individual octet is transmitted and received by the SPI or UART interface. See *AN706: EZSP-UART Host Interface Guide* and *AN711: EZSP-SPI Host Interface Guide*, for more information about the UART and SPI interfaces respectively.

2.3 Conceptual Overview

This section provides an overview of the concepts that are specific to EZSP or that differ from the EmberZNet PRO stack API. The commands and responses mentioned in this overview are described in more detail later in this document.

2.3.1 Stack Configuration

To ensure that the NCP and the host agree on the protocol format, the first command sent by the host after the NCP has reset must be the `version` command. There are a number of configuration values that affect the behavior of the stack. The host can read these values at any time using the `getConfigurationValue` command. After the NCP has reset, the host can modify any of the default values using the `setConfigurationValue` command. The host must then provide information about the application endpoints using the `addEndpoint` command.

The following table gives the minimum and maximum values for each of the configuration values. Also listed is the RAM cost—the number of bytes of additional RAM required to increase the configuration value by one. Because the total amount of RAM is fixed, the additional RAM required must be made available by reducing one of the other configuration values.

Note: Due to code size constraints, Silicon Labs does not bound check any EZSP values on the NCP. Silicon Labs recommends implementing bound checks on the host side.

Table 2-1. Configuration Values

Configuration Value	Min.	Max.	Units	RAM Cost	Description
EZSP_CONFIG_PACKET_BUFFER_COUNT	5	253	packet buffers	39	The number of packet buffers available to the stack. When set to the special value 0xFF, the NCP will allocate all remaining configuration RAM towards packet buffers, such that the resulting count will be the largest whole number of packet buffers that can fit into the available memory.
EZSP_CONFIG_NEIGHBOR_TABLE_SIZE	16	26	neighbors	18	The maximum number of router neighbors the stack can keep track of. A neighbor is a node within radio range.
EZSP_CONFIG_APS_UNICAST_MESSAGE_COUNT	0		messages	6	The maximum number of APS retried messages the stack can be transmitting at any time.
EZSP_CONFIG_BINDING_TABLE_SIZE	0	127	entries	2	The maximum number of non-volatile bindings supported by the stack.
EZSP_CONFIG_ADDRESS_TABLE_SIZE	0		entries	12	The maximum number of EUI64 to network address associations that the stack can maintain for the application. (Note: The total number of such address associations maintained by the NCP is the sum of the value of this setting and the value of ::EZSP_CONFIG_TRUST_CENTER_ADDRESS_CACHE_SIZE.).
EZSP_CONFIG_MULTICAST_TABLE_SIZE	0		entries	4	The maximum number of multicast groups that the device may be a member of.
EZSP_CONFIG_ROUTE_TABLE_SIZE	0		entries	6	The maximum number of destinations to which a node can route messages. This includes both messages originating at this node and those relayed for others.
EZSP_CONFIG_DISCOVERY_TABLE_SIZE	0		entries	10	The number of simultaneous route discoveries that a node will support.
EZSP_CONFIG_BROADCAST_ALARM_DATA_SIZE	0	16	bytes	1	The size of the alarm broadcast buffer.
EZSP_CONFIG_UNICAST_ALARM_DATA_SIZE (A)	0	16	bytes	(C)	The size of the unicast alarm buffers allocated for end device children.
EZSP_CONFIG_STACK_PROFILE	0			0	Specifies the stack profile.
EZSP_CONFIG_SECURITY_LEVEL	0	5		0	The security level used for security at the MAC and network layers. The supported values are 0 (no security) and 5 (payload is encrypted and a four-byte MIC is used for authentication).
EZSP_CONFIG_MAX_HOPS (B)	0		hops	0	The maximum number of hops for a message.
EZSP_CONFIG_MAX_END_DEVICE_CHILDREN (C)	0	64	children	9 + (A)	The maximum number of end device children that a router will support.
EZSP_CONFIG_INDIRECT_TRANSMISSION_TIMEOUT	0	30000	milli-seconds	0	The maximum amount of time that the MAC will hold a message for indirect transmission to a child.
EZSP_CONFIG_END_DEVICE_POLL_TIMEOUT	0	14	2^(D) seconds	0	The maximum amount of time that an end device child can wait between polls. If no poll is heard within this timeout, then the parent removes the end device from its tables. The timeout corresponding to a value of zero is 10 seconds. The timeout corresponding to a nonzero value N is 2^N minutes, ranging from 2^1 = 2 minutes to 2^14 = 16384 minutes.
EZSP_CONFIG_MOBILE_NODE_POLL_TIMEOUT	0		quarter seconds	0	The maximum amount of time that a mobile node can wait between polls. If no poll is heard within this timeout, then the parent removes the mobile node from its tables.
EZSP_CONFIG_RESERVED_MOBILE_CHILD_ENTRIES	0	(C)	entries	0	The number of child table entries reserved for use only by mobile nodes.
EZSP_CONFIG_TX_POWER_MODE	0	3		0	Enables boost power mode and/or the alternate transmitter output.
EZSP_CONFIG_DISABLE_RELAY	0	1		0	0: Allow this node to relay messages. 1: Prevent this node from relaying messages.

Configuration Value	Min.	Max.	Units	RAM Cost	Description
EZSP_CONFIG_TRUST_CENTER_ADDRESS_CACHE_SIZE	0		entries	12	The maximum number of EUI64 to network address associations that the Trust Center can maintain. These address cache entries are reserved for and reused by the Trust Center when processing device join/rejoin authentications. This cache size limits the number of overlapping joins the Trust Center can process within a narrow time window (e.g. two seconds), and thus should be set to the maximum number of near simultaneous joins the Trust Center is expected to accommodate. (Note: The total number of such address associations maintained by the NCP is the sum of the value of this setting and the value of ::EZSP_CONFIG_ADDRESS_TABLE_SIZE.)
EZSP_CONFIG_SOURCE_ROUTE_TABLE_SIZE	0		entries	4	The size of the source route table.
EZSP_CONFIG_FRAGMENT_WINDOW_SIZE	0	8	blocks	0	The number of blocks of a fragmented message that can be sent in a single window.
EZSP_CONFIG_FRAGMENT_DELAY_MS	0		milli-seconds	0	The time the stack will wait between sending blocks of a fragmented message.
EZSP_CONFIG_KEY_TABLE_SIZE	0		entries	4	The size of the Key Table used for storing individual link keys (if the device is a Trust Center) or Application Link Keys (if the device is a normal node).
EZSP_CONFIG_APS_ACK_TIMEOUT			milli-seconds	0	The APS ACK timeout value. The stack waits this amount of time between resends of APS retried messages.
EZSP_CONFIG_END_DEVICE_BIND_TIMEOUT	1		seconds	0	The time the coordinator will wait for a second end device bind request to arrive.
EZSP_CONFIG_PAN_ID_CONFLICT_REPORT_THRESHOLD	1	63	reports per minute	0	The number of PAN id conflict reports that must be received by the network manager within one minute to trigger a PAN id change.
EZSP_CONFIG_REQUEST_KEY_TIMEOUT	0	10	minutes	0	The timeout value in minutes for how long the Trust Center or a normal node waits for the ZigBee Request Key to complete. On the Trust Center this controls whether or not the device buffers the request, waiting for a matching pair of ZigBee Request Key. If the value is non-zero, the Trust Center buffers and waits for that amount of time. If the value is zero, the Trust Center does not buffer the request and immediately responds to the request. Zero is the most compliant behavior.
EZSP_CONFIG_CERTIFICATE_TABLE_SIZE	0	1		0	This value indicates the size of the runtime modifiable certificate table. Normally certificates are stored in MFG tokens but this table can be used to field upgrade devices with new Smart Energy certificates. This value cannot be set, it can only be queried.
EZSP_CONFIG_APPLICATION_ZDO_FLAGS	0	255		0	This is a bitmask that controls which incoming ZDO request messages are passed to the application. The bits are defined in the EmberZdoConfigurationFlags enumeration. To see if the application is required to send a ZDO response in reply to an incoming message, the application must check the APS options bitfield within the incomingMessageHandler callback to see if the EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED flag is set.
EZSP_CONFIG_BROADCAST_TABLE_SIZE	15	254	entries	6	The maximum number of broadcasts during a single broadcast timeout period.
EZSP_CONFIG_MAC_FILTER_TABLE_SIZE	0	254	entries	2	The size of the MAC filter list table.
EZSP_CONFIG_SUPPORTED_NETWORKS	1	2	entries	72	The number of supported networks.
EZSP_CONFIG_SEND_MULTICASTS_TO_SLEEPY_ADDRESS	0	1		0	Whether multicasts are sent to the RxOnWhenIdle=true address (0xFFFD) or the sleepy broadcast address (0xFFFF). The RxOnWhenIdle=true address is the ZigBee compliant destination for multicasts. 0=false, 1=true
EZSP_CONFIG_ZLL_GROUP_ADDRESSES	0	255		0	ZLL group address initial configuration.
EZSP_CONFIG_ZLL_RSSI_THRESHOLD	-128	127		0	ZLL RSSI threshold initial configuration.

Configuration Value	Min.	Max.	Units	RAM Cost	Description
EZSP_CONFIG_RF4CE_PAIRING_TABLE_SIZE	0	126	entries	48	The maximum number of pairings supported by the stack. Controllers must support at least one pairing table entry while targets must support at least five.
EZSP_CONFIG_RF4CE_PENDING_OUTGOING_PACKET_TABLE_SIZE	0	16	entries	16	The maximum number of outgoing RF4CE packets supported by the stack.
EZSP_CONFIG_MTORR_FLOW_CONTROL	0	1		0	Toggles the MTORR flow control in the stack. 0=false, 1=true
(Deprecated)EZSP_CONFIG_TRANSIENT_KEY_TIMEOUT_S	0	65535	seconds	0	The amount of time a trust center will store a transient key with which a device can use to join the network.

2.3.2 Policy Settings

There are some situations when the NCP must decide but there is not enough time to consult with the host. The host can control what decision is made by setting the policy in advance. The NCP will then make decisions according to the current policy. The host is informed via callbacks each time a decision is made, but by the time the news reaches the host, it is too late to change that decision. You can change the policies at any time by using the `setPolicy` command.

A policy is used for trust center behavior, external binding modification requests, unicast replies, generating `pollHandler` callbacks, and the contents of the `messageSent` callback.

2.3.3 Unicast Replies

The policy for unicast replies allows the host to decide whether it wants to supply the NCP with a reply payload for every retried unicast received. If the host sets the policy to not supply a reply, the NCP will automatically send an empty reply (containing no payload) for every retried unicast received. If the host sets the policy to supply the reply, then the NCP will only send a reply when instructed by the host.

If the reply does not reach the sender before the APS retry timeout expires, the sender will transmit the unicast again. The host must process the incoming message and supply the reply quickly enough to avoid retransmission by the sender. Provided this timing constraint is met, multiple unicasts can be received before the first reply is supplied and the replies can be supplied in any order.

2.3.4 SPI Interface Callbacks

Asynchronous callbacks from the NCP are sent to the host as the response to a `callback` command. The NCP uses the interrupt line to indicate that the host should send a `callback` command. The NCP will queue multiple callbacks while it waits for the host. Each response only delivers one callback. If the NCP receives the `callback` command when there are no pending callbacks, it will reply with the `noCallbacks` response.

2.3.5 UART Interface Callbacks

By default, callbacks from the NCP are sent to the host asynchronously as soon as they occur, and the host never needs to send the `callback` command. The host can disable asynchronous callbacks by setting `EZSP_VALUE_UART_SYNCH_CALLBACKS` to `1` using the `setValue` command. Callbacks will then only be sent to the host as the response to a callback command.

2.3.6 SPI Interface Power Management

The NCP always idles its processor whenever possible. To further reduce power consumption when connected using the SPI interface, the NCP can be put to sleep by the host. The UART interface is designed for gateway applications and does not support power management. In power down mode, only an external interrupt will wake the NCP. In deep sleep mode, the NCP will use its internal timer to wake up for scheduled events. The NCP provides two independent timers that the host can use for any purpose, including waking up the NCP from deep sleep mode. Timers are set using the `setTimer` command and generate `timerHandler` callbacks.

The frame control byte of every command tells the NCP which sleep mode to enter after it has responded to the command. Including this information in every command (instead of having a separate power management command) allows the NCP to be put to sleep faster. If the host needs to put the NCP to sleep without also performing another action, the `nop` command can be used.

In deep sleep mode, the NCP will wake up for an internal event. If the event does not produce a callback for the host, the NCP will go back to sleep once the event has been handled. If the event does produce a callback, the NCP will signal the host and remain awake waiting for the `callback` command. If the frame control byte of the `callback` command specifies deep sleep mode, then the NCP would normally go back to sleep after responding with the callback. However, if there is a second callback pending, the NCP will remain awake waiting for another `callback` command.

To avoid disrupting the operation of the network, only put the NCP to sleep when it is not joined to a network or when it is joined as a sleeping end device. If the NCP is joined as a sleeping end device, then it must poll its parent in order to receive messages. The host controls the polling behavior using the `pollForData` command. Polls are sent periodically with the interval set by the host or a single poll can be sent. The result of every poll attempt is optionally reported using the `pollCompleteHandler` callback.

2.3.7 Tokens

Some of the non-volatile storage on the NCP is made available for use by the host. Tokens stored in the Simulated EEPROM can be read and written using the `setToken` and `getToken` commands. Each token is 8 bytes. 32 tokens are available on EFR32 devices when using Simulated EEPROM v2, otherwise 8 tokens are available. Tokens preserve their values between reboots. The manufacturing tokens stored in the Flash Information Area can be read using the `getMfgToken` command.

2.3.8 NCP Status

The frame control byte of every response sent by the NCP contains four status fields:

- The overflow bit is set if the NCP ran out of memory at any time since the previous response was sent. If this bit is set, then messages may have been lost.
- The truncated bit is set if the NCP truncated the current response. If this bit is set, the command from the host produced a response larger than the maximum EZSP frame length.
- The callback pending bit is set if the NCP has one or more callbacks that have not been delivered to the host.
- The callback type field identifies a response as either an asynchronous callback (UART interface only), a synchronous callback, or not a callback.

You can use the `nop` command to check the status of the NCP without also performing another action.

2.3.9 Random Number Generator

The host can obtain a random number from the NCP using the `getRandomNumber` command. The random number is generated from analog noise in the radio and can be used to seed a random number generator on the host.

3 Protocol Format

Figure 3-1 illustrates the EZSP frame format.

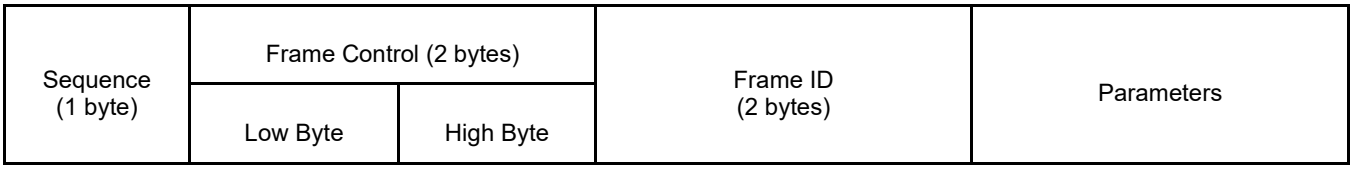

Figure 3-1. EZSP Frame Format

The first byte of all EZSP frames is a sequence number. The host should increment the sequence number each time a command is sent to the NCP. The response sent by the NCP uses the sequence number of the command, except when the response is a callback. Callback responses contain the sequence number of the last command seen at the time the callback occurred on the NCP. Starting with EZSP version 8, both Frame Control and Frame ID consist of two bytes. Table 3-1 shows a summary of the two-byte Frame Control. All Frame IDs are described in detail from section 4 Configuration Frames through section 17 Secure EZSP Frames.

The EZSP Version command is a special command. It is used by the Host to retrieve the EZSP version of the NCP to verify that the Host and NCP are working with the same EZSP version. An NCP will not accept other EZSP commands from the Host until after a Version command is successfully transacted. Because the EZSP frame format has evolved over software releases, the EZSP Version command must be interoperable between different EZSP versions. To support successful execution of a Version command between a Host and NCP that have different EZSP versions, the Version command additionally is executable using a legacy EZSP frame format. For practical purposes, the Version command should be executed using that legacy frame format.

Figure 3-2 illustrates the legacy frame format used for the EZSP Version command.

Figure 3-2. EZSP Version Command – Legacy Frame Format

- The single Frame Control byte corresponds to the Low Byte of the two-byte Frame Control of the regular frame format (for definitions of the bits).
- The 16-bit Version command code 0x0000 is shortened in the single-byte Frame ID of the Version legacy frame format to be 0x00.

Table 3-1. Frame Control Summary

Type	Table Number	Description
Frame Control Low Byte	Table 3-2	Frame control low byte
Frame Control Low Byte	Table 3-3	Sleep modes
Frame Control Low Byte	Table 3-4	Overflow status bit
Frame Control Low Byte	Table 3-5	Truncated status bit
Frame Control Low Byte	Table 3-6	Callback pending status bit
Frame Control Low Byte	Table 3-7	Callback types
Frame Control High Byte	Table 3-8	Extended frame control byte
Frame Control High Byte	Table 3-9	Security enabled status bit
Frame Control High Byte	Table 3-10	Padding enabled status bit
Frame Control High Byte	Table 3-11	Frame format version

Table 3-2. Frame Control Low Byte

Bit	Command	Response
7 (MSB)	0	1
6	networkIndex[1]	networkIndex[1]
5	networkIndex[0]	networkIndex[0]
4	0 (reserved)	callbackType[1]
3	0 (reserved)	callbackType[0]
2	0 (reserved)	callbackPending
1	sleepMode[1]	truncated
0 (LSB)	sleepMode[0]	overflow

Table 3-3. Sleep Modes

sleepMode[1]	sleepMode[0]	Description
1	1	Reserved
1	0	Power down
0	1	Deep sleep
0	0	Idle

Table 3-4. Overflow Status Bit

overflow	Description
1	The NCP ran out of memory since the previous response.
0	No memory shortage since the previous response.

Table 3-5. Truncated Status Bit

truncated	Description
1	The NCP truncated the current response to avoid exceeding the maximum EZSP frame length.
0	The current response was not truncated.

Table 3-6. Callback Pending Status Bit

callbackPending	Description
1	A callback is pending on the NCP. If this response is a callback, at least one more callback is available.
0	All callbacks have been delivered to the host.

Table 3-7. Callback Types

callbackType[1]	callbackType[0]	Description
1	1	Reserved.
1	0	(UART interface only) This response is an asynchronous callback. It was not sent in response to a callback command.
0	1	This response is a synchronous callback. It was sent in response to a callback command.
0	0	This response is not a callback.

Table 3-8. Extended Frame Control Byte

Bit	Command	Response
7 (MSB)	securityEnabled	securityEnabled
6	paddingEnabled	paddingEnabled
5	0 (reserved)	0 (reserved)
4	0 (reserved)	0 (reserved)
3	0 (reserved)	0 (reserved)
2	0 (reserved)	0 (reserved)
1	frameFormatVersion[1]	frameFormatVersion[1]
0 (LSB)	frameFormatVersion[0]	frameFormatVersion[0]

Table 3-9. Security Enabled Status Bit

securityEnabled	Description
1	Security is enabled.
0	Security is not enabled.

Table 3-10. Padding Enabled Status Bit

securityEnabled	Description
1	Padding is enabled.
0	Padding is not enabled.

Table 3-11. Frame Format Version

frameFormatVersion[1]	frameFormatVersion[0]	Description
1	1	Reserved
1	0	Reserved
0	1	Version 1
0	0	Version 0

3.1 Type Definitions

Type	Alias	Description
Bool	uint8_t	Boolean type with values true and false.
EzspConfigId	uint8_t	Identifies a configuration value.
EzspValueId	uint8_t	Identifies a value.
EzspExtendedValueId	uint8_t	Identifies a value based on specified characteristics. Each set of characteristics is unique to that value and is specified during the call to get the extended value.
EzspEndpointFlags	uint16_t	Flags associated with the endpoint data configured on the NCP.
EmberConfigTxPowerMode	uint16_t	Values for EZSP_CONFIG_TX_POWER_MODE.
EzspPolicyId	uint8_t	Identifies a policy.
EzspDecisionBitmask	uint16_t	This is the policy decision bitmask that controls the trust center decision strategies. The bitmask is modified and extracted from the EzspDecisionId for supporting bitmask operations.

Type	Alias	Description
EzspDecisionId	uint8_t	Identifies a policy decision.
EzspMfgTokenId	uint8_t	Manufacturing token IDs used by ezspGetMfgToken().
EzspStatus	uint8_t	Status values used by EZSP.
EmberStatus	uint8_t	Return type for stack functions.
EmberEventUnits	uint8_t	Either marks an event as inactive or specifies the units for the event execution time.
EmberNodeType	uint8_t	The type of the node.
EmberNetworkStatus	uint8_t	The possible join states for a node.
EmberIncomingMessageType	uint8_t	Incoming message types.
EmberOutgoingMessageType	uint8_t	Outgoing message types.
EmberMacPassthroughType	uint8_t	MAC passthrough message type flags.
EmberBindingType	uint8_t	Binding types.
EmberApsOption	uint16_t	Options to use when sending a message.
EzspNetworkScanType	uint8_t	Network scan types.
EmberJoinDecision	uint8_t	Decision made by the trust center when a node attempts to join.
EmberInitialSecurityBitmask	uint16_t	This is the Initial Security Bitmask that controls the use of various security features.
EmberCurrentSecurityBitmask	uint16_t	This is the Current Security Bitmask that details the use of various security features.
EmberKeyType	uint8_t	Describes the type of ZigBee security key.
EmberKeyStructBitmask	uint16_t	Describes the presence of valid data within the EmberKeyStruct structure.
EmberDeviceUpdate	uint8_t	The status of the device update.
EmberKeyStatus	uint8_t	The status of the attempt to establish a key.
EmberCounterType	uint8_t	Defines the events reported to the application by the *readAndClearCounters* command.
EmberJoinMethod	uint8_t	The type of method used for joining.
EmberZdoConfigurationFlags	uint8_t	Flags for controlling which incoming ZDO requests are passed to the application. To see if the application is required to send a ZDO response to an incoming message, the application must check the APS options bitfield within the incomingMessageHandler callback to see if the EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED flag is set.
EmberConcentratorType	uint16_t	Type of concentrator.
EmberZllState	uint16_t	ZLL device state identifier.
EmberZllKeyIndex	uint8_t	ZLL key encryption algorithm enumeration.
EzspZllNetworkOperation	uint8_t	Differentiates among ZLL network operations.
EzspSourceRouteOverheadInformation	uint8_t	Validates Source Route Overhead Information cached.
EmberNetworkInitBitmask	uint16_t	Bitmask options for emberNetworkInit().
EmberMultiPhyNwkConfig	uint8_t	Network configuration for the desired radio interface for multi-phy network.
EmberDutyCycleState	uint8_t	Duty cycle states.
EmberRadioPowerMode	uint8_t	Radio power modes.
EmberEntropySource	uint8_t	Entropy sources.
sl_zb_sec_man_key_type_t	uint8_t	Key types recognized by Zigbee Security Manager.
sl_zb_sec_man_derived_key_type_t	Uint16_t	Derived key types recognized by Zigbee Security Manager.

Type	Alias	Description
sl_zigbee_sec_man_flags_t	uint8_t	Flags for key operations.
EmberNodeId	uint16_t	16-bit ZigBee network address.
sl_status_t	uint32_t	See sl_status.h for an enumerated list.
EmberPanId	uint16_t	802.15.4 PAN ID.
EmberMulticastId	uint16_t	16-bit ZigBee multicast group identifier.
EmberEUI64	uint8_t[8]	EUI 64-bit ID (an IEEE address).
EmberDutyCycleHectoPct	uint16_t	The percent of duty cycle for a limit. Duty Cycle, Limits, and Thresholds are reported in units of Percent * 100 (i.e. 10000 = 100.00%, 1 = 0.01%).
EmberLibraryId	uint8_t	A library identifier
EmberLibraryStatus	uint8_t	The presence and status of the Ember library.
EmberGpSecurityLevel	uint8_t	The security level of the GPD.
EmberGpKeyType	uint8_t	The type of security key to use for the GPD.
EmberGpProxyTableEntryStatus	uint8_t	The proxy table entry status
EmberGpSecurityFrameCounter	uint32_t	The security frame counter
EmberGpSinkTableEntryStatus	uint8_t	The sink table entry status

3.2 Structure Definitions

Structure	Field	Description
EmberNetworkParameters		Network parameters.
	uint8_t[8] extendedPanId	The network's extended PAN identifier.
	uint16_t panId	The network's PAN identifier.
	uint8_t radioTxPower	A power setting, in dBm.
	uint8_t radioChannel	A radio channel.
	EmberJoinMethod joinMethod	The method used to initially join the network.
	EmberNodeId nwkManagerId	NWK Manager ID. The ID of the network manager in the current network. This may only be set at joining when using EMBER_USE_CONFIGURED_NWK_STATE as the join method.
	uint8_t nwkUpdateId	NWK Update ID. The value of the ZigBee nwkUpdateId known by the stack. This is used to determine the newest instance of the network after a PAN ID or channel change. This may only be set at joining when using EMBER_USE_CONFIGURED_NWK_STATE as the join method.
	uint32_t channels	NWK channel mask. The list of preferred channels that the NWK manager has told this device to use when searching for the network. This may only be set at joining when using EMBER_USE_CONFIGURED_NWK_STATE as the join method.
EmberMultiPhyRadioParameters		Radio parameters.
	int8_t radioTxPower	A power setting, in dBm.
	uint8_t radioPage	A radio page.
	uint8_t radioChannel	A radio channel.

Structure	Field	Description
EmberZigbeeNetwork		The parameters of a ZigBee network.
	uint8_t channel	The 802.15.4 channel associated with the network.
	uint16_t panId	The network's PAN identifier.
	uint8_t[8] extendedPanId	The network's extended PAN identifier.
	bool allowingJoin	Whether the network is allowing MAC associations.
	uint8_t stackProfile	The Stack Profile associated with the network.
	uint8_t nwkUpdateId	The instance of the Network.
EmberApsFrame		ZigBee APS frame parameters.
	uint16_t profileId	The application profile ID that describes the format of the message.
	uint16_t clusterId	The cluster ID for this message.
	uint8_t sourceEndpoint	The source endpoint.
	uint8_t destinationEndpoint	The destination endpoint.
	EmberApsOption options	A bitmask of options.
	uint16_t groupId	The group ID for this message, if it is multicast mode.
	uint8_t sequence	The sequence number.
EmberBindingTableEntry		An entry in the binding table.
	EmberBindingType type	The type of binding.
	uint8_t local	The endpoint on the local node.
	uint16_t clusterId	A cluster ID that matches one from the local endpoint's simple descriptor. This cluster ID is set by the provisioning application to indicate which part an endpoint's functionality is bound to this particular remote node and is used to distinguish between unicast and multicast bindings. Note that a binding can be used to send messages with any cluster ID, not just the one listed in the binding.
	uint8_t remote	The endpoint on the remote node (specified by identifier).
	EmberEUI64 identifier	A 64-bit identifier. This is either the destination EUI64 (for unicasts) or the 64-bit group address (for multicasts).
	uint8_t networkIndex	The index of the network the binding belongs to.
EmberMulticastTableEntry		A multicast table entry indicates that a particular endpoint is a member of a particular multicast group. Only devices with an endpoint in a multicast group will receive messages sent to that multicast group.
	EmberMulticastId multicastId	The multicast group ID.
	uint8_t endpoint	The endpoint that is a member, or 0 if this entry is not in use (the ZDO is not a member of any multicast groups.)
	uint8_t networkIndex	The network index of the network the entry is related to.
EmberKeyData		A 128-bit key.
	uint8_t[16] contents	The key data.
EmberCertificateData		The implicit certificate used in CBKE.
	uint8_t[48] contents	The certificate data.
EmberPublicKeyData		The public key data used in CBKE.

Structure	Field	Description
	uint8_t[22] contents	The public key data.
EmberPrivateKeyData		The private key data used in CBKE.
	uint8_t[21] contents	The private key data.
EmberSmacData		The Shared Message Authentication Code data used in CBKE.
	uint8_t[16] contents	The Shared Message Authentication Code data.
EmberSignatureData		An ECDSA signature
	uint8_t[42] contents	The signature data.
EmberCertificate283k1Data		The implicit certificate used in CBKE.
	uint8_t[74] contents	The 283k1 certificate data.
EmberPublicKey283k1Data		The public key data used in CBKE.
	uint8_t[37] contents	The 283k1 public key data.
EmberPrivateKey283k1Data		The private key data used in CBKE.
	uint8_t[36] contents	The 283k1 private key data.
EmberSignature283k1Data		An ECDSA signature
	uint8_t[72] contents	The 283k1 signature data.
EmberMessageDigest		The calculated digest of a message
	uint8_t[16] contents	The calculated digest of a message.
EmberAesMmoHashContext		The hash context for an ongoing hash operation.
	uint8_t[16] result	The result of ongoing the hash operation.
	uint32_t length	The total length of the data that has been hashed so far.
EmberBeaconData		Beacon data structure.
	uint8_t channel	The channel of the received beacon.
	uint8_t lqi	The LQI of the received beacon.
	int8_t rssi	The RSSI of the received beacon.
	uint8_t depth	The depth of the received beacon.
	uint8_t nwkUpdateId	The network update ID of the received beacon.
	int8_t power	The power level of the received beacon. This field is valid only if the beacon is an enhanced beacon.
	int8_t parentPriority	The TC connectivity and long uptime from capacity field.
	EmberPanId panId	The PAN ID of the received beacon.
	uint8_t[8] extendedPanId	The extended PAN ID of the received beacon.
	EmberNodeId sender	The sender of the received beacon.
	bool enhanced	Whether or not the beacon is enhanced.
	bool permitJoin	Whether the beacon is advertising permit join.
	bool hasCapacity	Whether the beacon is advertising capacity.
EmberBeaconIterator		Defines an iterator that is used to loop over cached beacons. Do not write to fields denoted as Private.
	EmberBeaconData beacon	The retrieved beacon.
	uint8_t index	(Private) The index of the retrieved beacon.

Structure	Field	Description
EmberBeaconClassificationParams		The parameters related to beacon prioritization.
	int8_t minRssiForReceivingPkts	The minimum RSSI value for receiving packets that is used in some beacon prioritization algorithms.
	uint16_t beaconClassificationMask	The beacon classification mask that identifies which beacon prioritization algorithm to pick and defines the relevant parameters.
EmberNeighborTableEntry		A neighbor table entry stores information about the reliability of RF links to and from neighboring nodes.
	uint16_t shortId	The neighbor's two-byte network id
	uint8_t averageLqi	An exponentially weighted moving average of the link quality values of incoming packets from this neighbor as reported by the PHY.
	uint8_t inCost	The incoming cost for this neighbor, computed from the average LQI. Values range from 1 for a good link to 7 for a bad link.
	uint8_t outCost	The outgoing cost for this neighbor, obtained from the most recently received neighbor exchange message from the neighbor. A value of zero means that a neighbor exchange message from the neighbor has not been received recently enough, or that our id was not present in the most recently received one.
	uint8_t age	The number of aging periods elapsed since a link status message was last received from this neighbor. The aging period is 16 seconds.
	EmberEUI64 longId	The 8-byte EUI64 of the neighbor.
EmberRouteTableEntry		A route table entry stores information about the next hop along the route to the destination.
	uint16_t destination	The short id of the destination. A value of 0xFFFF indicates the entry is unused.
	uint16_t nextHop	The short id of the next hop to this destination.
	uint8_t status	Indicates whether this entry is active (0), being discovered (1), unused (3), or validating (4).
	uint8_t age	The number of seconds since this route entry was last used to send a packet.
	uint8_t concentratorType	Indicates whether this destination is a High RAM Concentrator (2), a Low RAM Concentrator (1), or not a concentrator (0).
	uint8_t routeRecordState	For a High RAM Concentrator, indicates whether a route record is needed (2), has been sent (1), or is no long needed (0) because a source routed message from the concentrator has been received.
EmberInitialSecurityState		The security data used to set the configuration for the stack, or the retrieved configuration currently in use.
	EmberInitialSecurityBitmask bitmask	A bitmask indicating the security state used to indicate what the security configuration will be when the device forms or joins the network.
	EmberKeyData preconfiguredKey	The pre-configured Key data that should be used when forming or joining the network. The security bitmask must be set with the EMBER_HAVE_PRECONFIGURED_KEY bit to indicate that the key contains valid data.

Structure	Field	Description
	EmberKeyData networkKey	The Network Key that should be used by the Trust Center when it forms the network, or the Network Key currently in use by a joined device. The security bitmask must be set with EMBER_HAVE_NETWORK_KEY to indicate that the key contains valid data.
	uint8_t networkKeySequenceNumber	The sequence number associated with the network key. This is only valid if the EMBER_HAVE_NETWORK_KEY has been set in the security bitmask.
	EmberEUI64 preconfiguredTrustCenterEui64	This is the long address of the trust center on the network that will be joined. It is usually NOT set prior to joining the network and instead it is learned during the joining message exchange. This field is only examined if EMBER_HAVE_TRUST_CENTER_EUI64 is set in the EmberInitialSecurityState::bitmask. Most devices should clear that bit and leave this field alone. This field must be set when using commissioning mode.
EmberCurrentSecurityState		The security options and information currently used by the stack.
	EmberCurrentSecurityBitmask bitmask	A bitmask indicating the security options currently in use by a device joined in the network.
	EmberEUI64 trustCenterLongAddress	The IEEE Address of the Trust Center device.
EmberKeyStruct		A structure containing a key and its associated data.
	EmberKeyStructBitmask bitmask	A bitmask indicating the presence of data within the various fields in the structure.
	EmberKeyType type	The type of the key.
	EmberKeyData key	The actual key data.
	uint32_t outgoingFrameCounter	The outgoing frame counter associated with the key.
	uint32_t incomingFrameCounter	The frame counter of the partner device associated with the key.
	uint8_t sequenceNumber	The sequence number associated with the key.
	EmberEUI64 partnerEUI64	The IEEE address of the partner device also in possession of the key.
EmberNetworkInitStruct		Network Initialization parameters.
	EmberNetworkInitBitmask bitmask	Configuration options for network init.
EmberZllSecurityAlgorithmData		Data associated with the ZLL security algorithm.
	uint32_t transactionId	Transaction identifier.
	uint32_t responseId	Response identifier.
	uint16_t bitmask	Bitmask.
EmberZllNetwork		The parameters of a ZLL network.
	EmberZigbeeNetwork zigbeeNetwork	The parameters of a ZigBee network.
	EmberZllSecurityAlgorithmData securityAlgorithm	Data associated with the ZLL security algorithm.
	EmberEUI64 eui64	Associated EUI64.
	EmberNodeId nodeId	The node id.
	EmberZllState state	The ZLL state.
	EmberNodeType nodeType	The node type.

Structure	Field	Description
	uint8_t numberSubDevices	The number of sub devices.
	uint8_t totalGroupIdentifiers	The total number of group identifiers.
	uint8_t rssiCorrection	RSSI correction value.
EmberZllInitialSecurityState		Describes the initial security features and requirements that will be used when forming or joining ZLL networks.
	uint32_t bitmask	Unused bitmask; reserved for future use.
	EmberZllKeyIndex keyIndex	The key encryption algorithm advertised by the application.
	EmberKeyData encryptionKey	The encryption key for use by algorithms that require it.
	EmberKeyData preconfiguredKey	The pre-configured link key used during classical ZigBee commissioning.
EmberZllDeviceInfoRecord		Information about a specific ZLL Device.
	EmberEUI64 ieeeAddress	EUI64 associated with the device.
	uint8_t endpointId	Endpoint id.
	uint16_t profileId	Profile id.
	uint16_t deviceId	Device id.
	uint8_t version	Associated version.
	uint8_t groupIdCount	Number of relevant group ids.
EmberZllAddressAssignment		ZLL address assignment data.
	EmberNodeId nodeId	Relevant node id.
	EmberNodeId freeNodeIdMin	Minimum free node id.
	EmberNodeId freeNodeIdMax	Maximum free node id.
	EmberMulticastId groupIdMin	Minimum group id.
	EmberMulticastId groupIdMax	Maximum group id.
	EmberMulticastId freeGroupIdMin	Minimum free group id.
	EmberMulticastId freeGroupIdMax	Maximum free group id.
EmberTokTypeStackZllData		Public API for ZLL stack data token.
	uint32_t bitmask	Token bitmask.
	uint16_t freeNodeIdMin	Minimum free node id.
	uint16_t freeNodeIdMax	Maximum free node id.
	uint16_t myGroupIdMin	Local minimum group id.
	uint16_t freeGroupIdMin	Minimum free group id.
	uint16_t freeGroupIdMax	Maximum free group id.
	uint8_t rssiCorrection	RSSI correction value.
EmberTokTypeStackZllSecurity		Public API for ZLL stack security token.
	uint32_t bitmask	Token bitmask.
	uint8_t keyIndex	Key index.
	uint8_t[16] encryptionKey	Encryption key.
	uint8_t[16] preconfiguredKey	Preconfigured key.
EmberDutyCycleLimits		A structure containing duty cycle limit configurations. All limits are absolute, and are required to be as follows: suspLimit > critThresh > limitThresh For example: suspLimit = 250 (2.5%), critThresh = 180 (1.8%), limitThresh 100 (1.00%).

Structure	Field	Description
	uint16_t vendorId	The vendor identifier field shall contain the vendor identifier of the node.
	uint8_t[7] vendorString	The vendor string field shall contain the vendor string of the node.
EmberPerDeviceDutyCycle		A structure containing per device overall duty cycle consumed (up to the suspend limit).
	EmberNodeId nodeId	Node Id of device whose duty cycle is reported.
	EmberDutyCycleHectoPct dutyCycleConsumed	Amount of overall duty cycle consumed (up to suspend limit).
EmberTransientKeyData		The transient key data structure.
	EmberEUI64 eui64	The IEEE address paired with the transient link key.
	EmberKeyData keyData	The key data structure matching the transient key.
	EmberKeyStructBitmask bitmask	This bitmask indicates whether various fields in the structure contain valid data.
	uint16_t remainingTimeSeconds	The number of seconds remaining before the key is automatically timed out of the transient key table.
	uint8_t networkIndex	The network index indicates which NWK uses this key.
EmberChildData		A structure containing a child node's data.
	EmberEUI64 eui64	The EUI64 of the child
	EmberNodeType type	The node type of the child
	EmberNodeId id	The short address of the child
	uint8_t phy	The phy of the child
	uint8_t power	The power of the child
	uint8_t timeout	The timeout of the child
	EmberEUI64 gpdIeeeAddress	The GPD's EUI64.
	uint32_t sourceId	The GPD's source ID.
	uint8_t applicationId	The GPD Application ID.
	uint8_t endpoint	The GPD endpoint.
sl_zb_sec_man_key_t		A 128-bit key.
	uint8_t[16] key	The key data.
sl_zb_sec_man_context_t		Context for Zigbee Security Manager operations.
	sl_zb_sec_man_key_type_t core_key_type	The type of key being referenced.
	uint8_t key_index	The index of the referenced key.
	sl_zb_sec_man_derived_key_type_t derived_type	The type of key derivation operation to perform on a key.
	EmberEUI64 eui64	The EUI64 associated with this key.
	uint8_t multi_network_index	Multi-network index.
	sl_zigbee_sec_man_flags_t flags	Flag bitmask.
	uint32_t psa_key_alg_permission	Algorithm to use with this key (for PSA APIs)
sl_zb_sec_man_network_key_info_t		Metadata for network keys.
	bool network_key_set	Whether the current network key is set.
	bool alternate_network_key_set	Whether the alternate network key is set.

Structure	Field	Description
	uint8_t network_key_sequence_number	Current network key sequence number.
	uint8_t alt_network_key_sequence_number	Alternate network key sequence number.
	uint32_t network_key_frame_counter	Frame counter for the network key.
sl_zb_sec_man_aps_key_metadata_t		Metadata for APS link keys.
	EmberKeyStructBitmask bitmask	Bitmask of key properties
	uint32_t outgoing_frame_counter	Outgoing frame counter.
	uint32_t incoming_frame_counter	Incoming frame counter.
	uint16_t ttl_in_seconds	Remaining lifetime (for transient keys).
EmberGpAddress		A GP address structure.
	uint8_t[8] id	Contains either a 4-byte source ID or an 8-byte IEEE address, as indicated by the value of the applicationId field.
	uint8_t applicationId	The GPD Application ID specifying either source ID (0x00) or IEEE address (0x02).
	uint8_t endpoint	The GPD endpoint.
EmberGpProxyTableEntry		The internal representation of a proxy table entry
	EmberGpProxyTableEntryStatus status	Internal status of the proxy table entry.
	uint32_t options	The tunneling options (this contains both options and extendedOptions from the spec).
	EmberGpAddress gpd	The addressing info of the GPD.
	EmberNodeId assignedAlias	The assigned alias for the GPD.
	uint8_t securityOptions	The security options field.
	EmberGpSecurityFrameCounter gpdSecurityFrameCounter	The security frame counter of the GPD.
	EmberKeyData gpdKey	The key to use for GPD.
	EmberGpSinkListEntry sinkList[GP_SINK_LIST_ENTRIES]	The list of sinks (hardcoded to 2 which is the spec minimum).
	uint8_t groupcastRadius	The groupcast radius.
	uint8_t searchCounter	The search counter.
EmberGpSinkTableEntry		The internal representation of a sink table entry.
	EmberGpSinkTableEntryStatus status	Internal status of the sink table entry.
	uint32_t options	The tunneling options (this contains both options and extendedOptions from the spec).
	EmberGpAddress gpd	The addressing info of the GPD.
	uint8_t deviceId	The device id for the GPD.
	EmberGpSinkListEntry sinkList[GP_SINK_LIST_ENTRIES]	The list of sinks (hardcoded to 2 which is the spec minimum).
	EmberNodeId assignedAlias	The assigned alias for the GPD.
	uint8_t groupcastRadius	The groupcast radius.
	uint8_t securityOptions	The security options field.
	EmberGpSecurityFrameCounter gpdSecurityFrameCounter	The security frame counter of the GPD.
	EmberKeyData gpdKey	The key to use for GPD.

Structure	Field	Description
EmberTokenInfo		Information of a token in the token table.
	uint32_t nvm3Key	NVM3 key of the token
	bool isCnt	Token is a counter type
	bool isIdx	Token is an indexed token
	uint8_t size	Size of the token
	uint8_t arraySize	Array size of the token
EmberTokenData		Token Data
	uint32_t size	Token data size in bytes
	uint8_t[64] data	Token data pointer

3.3 Named Values

bool	Value	Description
false	0x00	An alias for zero, used for clarity.
true	0x01	An alias for one, used for clarity.

EzspConfigId	Value	Description
EZSP_CONFIG_PACKET_BUFFER_COUNT	0x01	The NCP no longer supports configuration of packet buffer count at runtime using this parameter. Packet buffers must be configured using the EMBER_PACKET_BUFFER_COUNT macro when building the NCP project.
EZSP_CONFIG_NEIGHBOR_TABLE_SIZE	0x02	The maximum number of router neighbors the stack can keep track of. A neighbor is a node within radio range.
EZSP_CONFIG_APS_UNICAST_MESSAGE_COUNT	0x03	The maximum number of APS retried messages the stack can be transmitting at any time.
EZSP_CONFIG_BINDING_TABLE_SIZE	0x04	The maximum number of non-volatile bindings supported by the stack.
EZSP_CONFIG_ADDRESS_TABLE_SIZE	0x05	The maximum number of EUI64 to network address associations that the stack can maintain for the application. (Note, the total number of such address associations maintained by the NCP is the sum of the value of this setting and the value of ::EZSP_CONFIG_TRUST_CENTER_ADDRESS_CACHE_SIZE.).
EZSP_CONFIG_MULTICAST_TABLE_SIZE	0x06	The maximum number of multicast groups that the device may be a member of.
EZSP_CONFIG_ROUTE_TABLE_SIZE	0x07	The maximum number of destinations to which a node can route messages. This includes both messages originating at this node and those relayed for others.
EZSP_CONFIG_DISCOVERY_TABLE_SIZE	0x08	The number of simultaneous route discoveries that a node will support.
EZSP_CONFIG_STACK_PROFILE	0x0C	Specifies the stack profile.
EZSP_CONFIG_SECURITY_LEVEL	0x0D	The security level used for security at the MAC and network layers. The supported values are 0 (no security) and 5 (payload is encrypted and a four-byte MIC is used for authentication).
EZSP_CONFIG_MAX_HOPS	0x10	The maximum number of hops for a message.
EZSP_CONFIG_MAX_END_DEVICE_CHILDREN	0x11	The maximum number of end device children that a router will support.

EzspConfigId	Value	Description
EZSP_CONFIG_INDIRECT_TRANSMISSION_TIMEOUT	0x12	The maximum amount of time that the MAC will hold a message for indirect transmission to a child.
EZSP_CONFIG_END_DEVICE_POLL_TIMEOUT	0x13	The maximum amount of time that an end device child can wait between polls. If no poll is heard within this timeout, then the parent removes the end device from its tables. Value range 0-14. The timeout corresponding to a value of zero is 10 seconds. The timeout corresponding to a nonzero value N is 2^N minutes, ranging from 2^1 = 2 minutes to 2^14 = 16384 minutes.
EZSP_CONFIG_TX_POWER_MODE	0x17	Enables boost power mode and/or the alternate transmitter output.
EZSP_CONFIG_DISABLE_RELAY	0x18	0: Allow this node to relay messages. 1: Prevent this node from relaying messages.
EZSP_CONFIG_TRUST_CENTER_ADDRESS_CACHE_SIZE	0x19	The maximum number of EUI64 to network address associations that the Trust Center can maintain. These address cache entries are reserved for and reused by the Trust Center when processing device join/rejoin authentications. This cache size limits the number of overlapping joins the Trust Center can process within a narrow time window (e.g. two seconds), and thus should be set to the maximum number of near simultaneous joins the Trust Center is expected to accommodate. (Note, the total number of such address associations maintained by the NCP is the sum of the value of this setting and the value of ::EZSP_CONFIG_ADDRESS_TABLE_SIZE.)
EZSP_CONFIG_SOURCE_ROUTE_TABLE_SIZE	0x1A	The size of the source route table.
EZSP_CONFIG_FRAGMENT_WINDOW_SIZE	0x1C	The number of blocks of a fragmented message that can be sent in a single window.
EZSP_CONFIG_FRAGMENT_DELAY_MS	0x1D	The time the stack will wait (in milliseconds) between sending blocks of a fragmented message.
EZSP_CONFIG_KEY_TABLE_SIZE	0x1E	The size of the Key Table used for storing individual link keys (if the device is a Trust Center) or Application Link Keys (if the device is a normal node).
EZSP_CONFIG_APS_ACK_TIMEOUT	0x1F	The APS ACK timeout value. The stack waits this amount of time between resends of APS retried messages.
EZSP_CONFIG_BEACON_JITTER_DURATION	0x20	The duration of a beacon jitter, in the units used by the 15.4 scan parameter (((1 << duration) + 1) * 15ms), when responding to a beacon request.
EZSP_CONFIG_PAN_ID_CONFLICT_REPORT_THRESHOLD	0x22	The number of PAN id conflict reports that must be received by the network manager within one minute to trigger a PAN id change.
EZSP_CONFIG_REQUEST_KEY_TIMEOUT	0x24	The timeout value in minutes for how long the Trust Center or a normal node waits for the ZigBee Request Key to complete. On the Trust Center this controls whether or not the device buffers the request, waiting for a matching pair of ZigBee Request Key. If the value is non-zero, the Trust Center buffers and waits for that amount of time. If the value is zero, the Trust Center does not buffer the request and immediately responds to the request. Zero is the most compliant behavior.
EZSP_CONFIG_CERTIFICATE_TABLE_SIZE	0x29	This value indicates the size of the runtime modifiable certificate table. Normally certificates are stored in MFG tokens but this table can be used to field upgrade devices with new Smart Energy certificates. This value cannot be set, it can only be queried.

EzspConfigId	Value	Description
EZSP_CONFIG_APPLICATION_ZDO_FLAGS	0x2A	This is a bitmask that controls which incoming ZDO request messages are passed to the application. The bits are defined in the EmberZdoConfigurationFlags enumeration. To see if the application is required to send a ZDO response in reply to an incoming message, the application must check the APS options bitfield within the incomingMessageHandler callback to see if the EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED flag is set.
EZSP_CONFIG_BROADCAST_TABLE_SIZE	0x2B	The maximum number of broadcasts during a single broadcast timeout period.
EZSP_CONFIG_MAC_FILTER_TABLE_SIZE	0x2C	The size of the MAC filter list table.
EZSP_CONFIG_SUPPORTED_NETWORKS	0x2D	The number of supported networks.
EZSP_CONFIG_SEND_MULTICASTS_TO_SLEEPY_ADDRESS	0x2E	Whether multicasts are sent to the RxOnWhenIdle=true address (0xFFFD) or the sleepy broadcast address (0xFFFF). The RxOnWhenIdle=true address is the ZigBee compliant destination for multicasts.
EZSP_CONFIG_ZLL_GROUP_ADDRESSES	0x2F	ZLL group address initial configuration.
EZSP_CONFIG_ZLL_RSSI_THRESHOLD	0x30	ZLL rssi threshold initial configuration.
EZSP_CONFIG_MTORR_FLOW_CONTROL	0x33	Toggles the MTORR flow control in the stack.
EZSP_CONFIG_RETRY_QUEUE_SIZE	0x34	Setting the retry queue size. Applies to all queues. Default value in the sample applications is 16.
EZSP_CONFIG_NEW_BROADCAST_ENTRY_THRESHOLD	0x35	Setting the new broadcast entry threshold. The number(BROADCAST_TABLE_SIZE - NEW_BROADCAST_ENTRY_THRESHOLD) of broadcast table entries are reserved for relaying the broadcast messages originated on other devices. The local device will fail to originate a broadcast message after this threshold is reached. Setting this value to BROADCAST_TABLE_SIZE and greater will effectively kill this limitation.
(Deprecated) EZSP_CONFIG_TRANSIENT_KEY_TIMEOUT_S	0x36	The length of time, in seconds, that a trust center will store a transient link key that a device can use to join its network. A transient key is added with a call to emberAddTransientLinkKey. After the transient key is added, it will be removed once this amount of time has passed. A joining device will not be able to use that key to join until it is added again on the trust center. The default value is 300 seconds, i.e., 5 minutes.
EZSP_CONFIG_BROADCAST_MIN_ACKS_NEEDED	0x37	The number of passive acknowledgements to record from neighbors before we stop re-transmitting broadcasts
EZSP_CONFIG_TC_REJOINS_USING_WELL_KNOWN_KEY_TIMEOUT_S	0x38	The length of time, in seconds, that a trust center will allow a Trust Center (insecure) rejoin for a device that is using the well-known link key. This timeout takes effect once rejoins using the well-known key has been allowed. This command updates the sli_zigbee_allow_tc_rejoins_using_well_known_key_timeout_sec value.
EZSP_CONFIG_CTUNE_VALUE	0x39	Valid range of a CTUNE value is 0x0000-0x01FF. Higher order bits (0xFE00) of the 16-bit value are ignored.
EZSP_CONFIG_ASSUME_TC_CONCENTRATOR_TYPE	0x40	To configure non trust center node to assume a concentrator type of the trust center it join to, until it receive many-to-one route request from the trust center. For the trust center node, concentrator type is

EzspConfigId	Value	Description
		configured from the concentrator plugin. The stack by default assumes trust center be a low RAM concentrator that make other devices send route record to the trust center even without receiving a many-to-one route request. The default concentrator type can be changed by setting appropriate EmberAssumeTrustCenterConcentratorType config value.
EZSP_CONFIG_GP_PROXY_TABLE_SIZE	0x41	This is green power proxy table size. This value is read-only and cannot be set at runtime
EZSP_CONFIG_GP_SINK_TABLE_SIZE	0x42	This is green power sink table size. This value is read-only and cannot be set at runtime

EzspValueId	Value	Description
EZSP_VALUE_TOKEN_STACK_NODE_DATA	0x00	The contents of the node data stack token.
EZSP_VALUE_MAC_PASSTHROUGH_FLAGS	0x01	The types of MAC passthrough messages that the host wishes to receive.
EZSP_VALUE_EMBERNET_PASSTHROUGH_SOURCE_ADDRESS	0x02	The source address used to filter legacy EmberNet messages when the EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE flag is set in EZSP_VALUE_MAC_PASSTHROUGH_FLAGS.
EZSP_VALUE_FREE_BUFFERS	0x03	The number of available internal RAM general purpose buffers. Read only.
EZSP_VALUE_UART_SYNCH_CALLBACKS	0x04	Selects sending synchronous callbacks in ezsp-uart.
EZSP_VALUE_MAXIMUM_INCOMING_TRANSFER_SIZE	0x05	The maximum incoming transfer size for the local node. Default value is set to 82 and does not use fragmentation. Sets the value in Node Descriptor. To set, this takes the input of a uint8 array of length 2 where you pass the lower byte at index 0 and upper byte at index 1.
EZSP_VALUE_MAXIMUM_OUTGOING_TRANSFER_SIZE	0x06	The maximum outgoing transfer size for the local node. Default value is set to 82 and does not use fragmentation. Sets the value in Node Descriptor. To set, this takes the input of a uint8 array of length 2 where you pass the lower byte at index 0 and upper byte at index 1.
EZSP_VALUE_STACK_TOKEN_WRITING	0x07	A bool indicating whether stack tokens are written to persistent storage as they change.
EZSP_VALUE_STACK_IS_PERFORMING_REJOIN	0x08	A read-only value indicating whether the stack is currently performing a rejoin.
EZSP_VALUE_MAC_FILTER_LIST	0x09	A list of EmberMacFilterMatchData values.
EZSP_VALUE_EXTENDED_SECURITY_BITMASK	0x0A	The Ember Extended Security Bitmask.
EZSP_VALUE_NODE_SHORT_ID	0x0B	The node short ID.
EZSP_VALUE_DESCRIPTOR_CAPABILITY	0x0C	The descriptor capability of the local node. Write only.
EZSP_VALUE_STACK_DEVICE_REQUEST_SEQUENCE_NUMBER	0x0D	The stack device request sequence number of the local node.
EZSP_VALUE_RADIO_HOLD_OFF	0x0E	Enable or disable radio hold-off.
EZSP_VALUE_ENDPOINT_FLAGS	0x0F	The flags field associated with the endpoint data.
EZSP_VALUE_MFG_SECURITY_CONFIG	0x10	Enable/disable the Mfg security config key settings.
EZSP_VALUE_VERSION_INFO	0x11	Retrieves the version information from the stack on the NCP.

EzspValueId	Value	Description
EZSP_VALUE_NEXT_HOST_REJOIN_REASON	0x12	This will get/set the rejoin reason noted by the host for a subsequent call to emberFindAndRejoinNetwork(). After a call to emberFindAndRejoinNetwork() the host's rejoin reason will be set to EMBER_REJOIN_REASON_NONE. The NCP will store the rejoin reason used by the call to emberFindAndRejoinNetwork().Application is not required to do anything with this value. The App Framework sets this for cases of emberFindAndRejoinNetwork that it initiates, but if the app is invoking a rejoin directly, it should/can set this value to aid in debugging of any rejoin state machine issues over EZSP logs after the fact. The NCP doesn't do anything with this value other than cache it so you can read it later.
EZSP_VALUE_LAST_REJOIN_REASON	0x13	This is the reason that the last rejoin took place. This value may only be retrieved, not set. The rejoin may have been initiated by the stack (NCP) or the application (host). If a host initiated a rejoin the reason will be set by default to EMBER_REJOIN_DUE_TO_APP_EVENT_1. If the application wishes to denote its own rejoin reasons it can do so by calling ezspSetValue(EMBER_VALUE_HOST_REJOIN_REASON, EMBER_REJOIN_DUE_TO_APP_EVENT_X). X is a number corresponding to one of the app events defined. If the NCP initiated a rejoin it will record this value internally for retrieval by ezspGetValue(EZSP_VALUE_REAL_REJOIN_REASON).
EZSP_VALUE_NEXT_ZIGBEE_SEQUENCE_NUMBER	0x14	The next ZigBee sequence number.
EZSP_VALUE_CCA_THRESHOLD	0x15	CCA energy detect threshold for radio.
EZSP_VALUE_SET_COUNTER_THRESHOLD	0x17	The threshold value for a counter
EZSP_VALUE_RESET_COUNTER_THRESHOLDS	0x18	Resets all counters thresholds to 0xFF
EZSP_VALUE_CLEAR_COUNTERS	0x19	Clears all the counters
EZSP_VALUE_CERTIFICATE_283K1	0x1A	The node's new certificate signed by the CA.
EZSP_VALUE_PUBLIC_KEY_283K1	0x1B	The Certificate Authority's public key.
EZSP_VALUE_PRIVATE_KEY_283K1	0x1C	The node's new static private key.
EZSP_VALUE_NWK_FRAME_COUNTER	0x23	The NWK layer security frame counter value
EZSP_VALUE_APS_FRAME_COUNTER	0x24	The APS layer security frame counter value. Managed by the stack. Users should not set these unless doing backup and restore.
EZSP_VALUE_RETRY_DEVICE_TYPE	0x25	Sets the device type to use on the next rejoin using device type
EZSP_VALUE_ENABLE_R21_BEHAVIOR	0x29	Setting this byte enables R21 behavior on the NCP.
EZSP_VALUE_ANTENNA_MODE	0x30	Configure the antenna mode(0-don't switch,1-primary,2-secondary,3-TX antenna diversity).
EZSP_VALUE_ENABLE_PTA	0x31	Enable or disable packet traffic arbitration.
EZSP_VALUE_PTA_OPTIONS	0x32	Set packet traffic arbitration configuration options.
EZSP_VALUE_MFGLIB_OPTIONS	0x33	Configure manufacturing library options (0-non-CSMA transmits,1-CSMA transmits). To be used with Manufacturing library.

EzspValueId	Value	Description
EZSP_VALUE_USE_NEGOTIATED_POWER_BY_LPD	0x34	Sets the flag to use either negotiated power by link power delta (LPD) or fixed power value provided by user while forming/joining a network for packet transmissions on sub-ghz interface. This is mainly for testing purposes.
EZSP_VALUE_PTA_PWM_OPTIONS	0x35	Set packet traffic arbitration PWM options.
EZSP_VALUE_PTA_DIRECTIONAL_PRIORITY_PULSE_WIDTH	0x36	Set packet traffic arbitration directional priority pulse width in microseconds.
EZSP_VALUE_PTA_PHY_SELECT_TIMEOUT	0x37	Set packet traffic arbitration phy select timeout(ms).
EZSP_VALUE_ANTENNA_RX_MODE	0x38	Configure the RX antenna mode: (0-do not switch; 1-primary; 2-secondary; 3-RX antenna diversity).
EZSP_VALUE_NWK_KEY_TIMEOUT	0x39	Configure the timeout to wait for the network key before failing a join. Acceptable timeout range [3,255]. Value is in seconds.
EZSP_VALUE_FORCE_TX_AFTER_FAILED_CCA_ATTEMPTS	0x3A	The number of failed CSMA attempts due to failed CCA made by the MAC before continuing transmission with CCA disabled. This is the same as calling the emberForceTxAfterFailedCca(uint8_t csmaAttempts) API. A value of 0 disables the feature.
EZSP_VALUE_TRANSIENT_KEY_TIMEOUT_S	0x3B	The length of time, in seconds, that a trust center will store a transient link key that a device can use to join its network. A transient key is added with a call to sl_zb_sec_man_import_transient_key. After the transient key is added, it will be removed once this amount of time has passed. A joining device will not be able to use that key to join until it is added again on the trust center. The default value is 300 seconds (5 minutes).
EZSP_VALUE_COULOMB_COUNTER_USAGE	0x3C	Cumulative energy usage metric since the last value reset of the coulomb counter plugin. Setting this value will reset the coulomb counter.
EZSP_VALUE_MAX_BEACONS_TO_STORE	0x3D	When scanning, configure the maximum number of beacons to store in cache. Each beacon consumes one packet buffer in RAM.
EZSP_VALUE_END_DEVICE_TIMEOUT_OPTIONS_MASK	0x3E	Set the mask to filter out unacceptable child timeout options on a router.
EZSP_VALUE_END_DEVICE_KEEP_ALIVE_SUPPORT_MODE	0x3F	The end device keep-alive mode supported by the parent.
EZSP_VALUE_ACTIVE_RADIO_CONFIG	0x41	Return the active radio config. Read only. Values are 0: Default, 1: Antenna Diversity, 2: Co-Existence, 3: Antenna diversity and Co-Existence.
EZSP_VALUE_NWK_OPEN_DURATION	0x42	Return the number of seconds the network will remain open. A return value of 0 indicates that the network is closed. Read only.
EZSP_VALUE_TRANSIENT_DEVICE_TIMEOUT	0x43	Timeout in milliseconds to store entries in the transient device table. If the devices are not authenticated before the timeout, the entry shall be purged
EZSP_VALUE_KEY_STORAGE_VERSION	0x44	Return information about the key storage on an NCP. Returns 0 if keys are in classic key storage, and 1 if they are located in PSA key storage. Read only.
EZSP_VALUE_DELAYED_JOIN_ACTIVATION	0x45	Return activation state about TC Delayed Join on an NCP. A return value of 0 indicates that the feature is not activated.

EzspExtendedValueId	Value	Description
EZSP_EXTENDED_VALUE_ENDPOINT_FLAGS	0x00	The flags field associated with the specified endpoint.
EZSP_EXTENDED_VALUE_LAST_LEAVE_REASON	0x01	This is the reason for the node to leave the network as well as the device that told it to leave. The leave reason is the 1st byte of the value while the node ID is the 2nd and 3rd byte. If the leave was caused due to an API call rather than an over the air message, the node ID will be EMBER_UNKNOWN_NODE_ID (0xFFFD).
EZSP_EXTENDED_VALUE_GET_SOURCE_ROUTE_OVERHEAD	0x02	This number of bytes of overhead required in the network frame for source routing to a particular destination.

EzspEndpointFlags	Value	Description
EZSP_ENDPOINT_DISABLED	0x00	Indicates that the endpoint is disabled and NOT discoverable via ZDO.
EZSP_ENDPOINT_ENABLED	0x01	Indicates that the endpoint is enabled and discoverable via ZDO.

EmberConfigTxPowerMode	Value	Description
EMBER_TX_POWER_MODE_DEFAULT	0x00	Normal power mode and bi-directional RF transmitter output.
EMBER_TX_POWER_MODE_BOOST	0x01	Enable boost power mode. This is a high-performance radio mode which offers increased receive sensitivity and transmit power at the cost of an increase in power consumption.
EMBER_TX_POWER_MODE_ALTERNATE	0x02	Enable the alternate transmitter output. This allows for simplified connection to an external power amplifier via the RF_TX_ALT_P and RF_TX_ALT_N pins.
EMBER_TX_POWER_MODE_BOOST_AND_ALTERNATE	0x03	Enable both boost mode and the alternate transmitter output.

EzspPolicyId	Value	Description
EZSP_TRUST_CENTER_POLICY	0x00	Controls trust center behavior.
EZSP_BINDING_MODIFICATION_POLICY	0x01	Controls how external binding modification requests are handled.
EZSP_UNICAST_REPLIES_POLICY	0x02	Controls whether the Host supplies unicast replies.
EZSP_POLL_HANDLER_POLICY	0x03	Controls whether pollHandler callbacks are generated.
EZSP_MESSAGE_CONTENTS_IN_CALLBACK_POLICY	0x04	Controls whether the message contents are included in the messageSentHandler callback.
EZSP_TC_KEY_REQUEST_POLICY	0x05	Controls whether the Trust Center will respond to Trust Center link key requests.
EZSP_APP_KEY_REQUEST_POLICY	0x06	Controls whether the Trust Center will respond to application link key requests.
EZSP_PACKET_VALIDATE_LIBRARY_POLICY	0x07	Controls whether ZigBee packets that appear invalid are automatically dropped by the stack. A counter will be incremented when this occurs.
EZSP_ZLL_POLICY	0x08	Controls whether the stack will process ZLL messages.

EzspPolicyId	Value	Description
EZSP_TC_REJOINS_USING_WELL_KNOWN_KEY_POLICY	0x09	Controls whether Trust Center (insecure) rejoins for devices using the well-known link key are accepted. If rejoining using the well-known key is allowed, it is disabled again after sli_zigbee_allow_tc_rejoins_using_well_known_key_timeout_sec seconds.

EzspDecisionBitmask	Value	Description
EZSP_DECISION_BITMASK_DEFAULT_CONFIGURATION	0x0000	Disallow joins and rejoins.
EZSP_DECISION_ALLOW_JOINS	0x0001	Send the network key to all joining devices.
EZSP_DECISION_ALLOW_UNSECURED_REJOINS	0x0002	Send the network key to all rejoining devices.
EZSP_DECISION_SEND_KEY_IN_CLEAR	0x0004	Send the network key in the clear.
EZSP_DECISION_IGNORE_UNSECURED_REJOINS	0x0008	Do nothing for unsecured rejoins.
EZSP_DECISION_JOINS_USE_INSTALL_CODE_KEY	0x0010	Allow joins if there is an entry in the transient key table.
EZSP_DECISION_DEFER_JOINS	0x0020	Delay sending the network key to a new joining device.

EzspDecisionId	Value	Description
EZSP_DEFER_JOINS_REJOINS_HAVE_LINK_KEY	0x07	Delay sending the network key to a new joining device.
EZSP_DISALLOW_BINDING_MODIFICATION	0x10	EZSP_BINDING_MODIFICATION_POLICY default decision. Do not allow the local binding table to be changed by remote nodes.
EZSP_ALLOW_BINDING_MODIFICATION	0x11	EZSP_BINDING_MODIFICATION_POLICY decision. Allow remote nodes to change the local binding table.
EZSP_CHECK_BINDING_MODIFICATIONS_ARE_VALID_ENDPOINT_CLUSTERS	0x12	EZSP_BINDING_MODIFICATION_POLICY decision. Allows remote nodes to set local binding entries only if the entries correspond to endpoints defined on the device, and for output clusters bound to those endpoints.
EZSP_HOST_WILL_NOT_SUPPLY_REPLY	0x20	EZSP_UNICAST_REPLIES_POLICY default decision. The NCP will automatically send an empty reply (containing no payload) for every unicast received.
EZSP_HOST_WILL_SUPPLY_REPLY	0x21	EZSP_UNICAST_REPLIES_POLICY decision. The NCP will only send a reply if it receives a sendReply command from the Host.
EZSP_POLL_HANDLER_IGNORE	0x30	EZSP_POLL_HANDLER_POLICY default decision. Do not inform the Host when a child polls.
EZSP_POLL_HANDLER_CALLBACK	0x31	EZSP_POLL_HANDLER_POLICY decision. Generate a pollHandler callback when a child polls.
EZSP_MESSAGE_TAG_ONLY_IN_CALLBACK	0x40	EZSP_MESSAGE_CONTENTS_IN_CALLBACK_POLICY default decision. Include only the message tag in the messageSentHandler callback.
EZSP_MESSAGE_TAG_AND_CONTENTS_IN_CALLBACK	0x41	EZSP_MESSAGE_CONTENTS_IN_CALLBACK_POLICY decision. Include both the message tag and the message contents in the messageSentHandler callback.

EzspDecisionId	Value	Description
EZSP_DENY_TC_KEY_REQUESTS	0x50	EZSP_TC_KEY_REQUEST_POLICY decision. When the Trust Center receives a request for a Trust Center link key, it will be ignored.
EZSP_ALLOW_TC_KEY_REQUESTS_AND_SEND_CURRENT_KEY	0x51	EZSP_TC_KEY_REQUEST_POLICY decision. When the Trust Center receives a request for a Trust Center link key, it will reply to it with the corresponding key.
EZSP_ALLOW_TC_KEY_REQUEST_AND_GENERATE_NEW_KEY	0x52	EZSP_TC_KEY_REQUEST_POLICY decision. When the Trust Center receives a request for a Trust Center link key, it will generate a key to send to the joiner. After generation, the key will be added to the transient key table and after verification this key will be added to the link key table.
EZSP_DENY_APP_KEY_REQUESTS	0x60	EZSP_APP_KEY_REQUEST_POLICY decision. When the Trust Center receives a request for an application link key, it will be ignored.
EZSP_ALLOW_APP_KEY_REQUESTS	0x61	EZSP_APP_KEY_REQUEST_POLICY decision. When the Trust Center receives a request for an application link key, it will randomly generate a key and send it to both partners.
EZSP_PACKET_VALIDATE_LIBRARY_CHECKS_ENABLED	0x62	Indicates that packet validate library checks are enabled on the NCP.
EZSP_PACKET_VALIDATE_LIBRARY_CHECKS_DISABLED	0x63	Indicates that packet validate library checks are NOT enabled on the NCP.

EzspMfgTokenId	Value	Description
EZSP_MFG_CUSTOM_VERSION	0x00	Custom version (2 bytes).
EZSP_MFG_STRING	0x01	Manufacturing string (16 bytes).
EZSP_MFG_BOARD_NAME	0x02	Board name (16 bytes).
EZSP_MFG_MANUF_ID	0x03	Manufacturing ID (2 bytes).
EZSP_MFG_PHY_CONFIG	0x04	Radio configuration (2 bytes).
EZSP_MFG_BOOTLOAD_AES_KEY	0x05	Bootload AES key (16 bytes).
EZSP_MFG_ASH_CONFIG	0x06	ASH configuration (40 bytes).
EZSP_MFG_EZSP_STORAGE	0x07	EZSP storage (8 bytes).
EZSP_STACK_CAL_DATA	0x08	Radio calibration data (64 bytes). 4 bytes are stored for each of the 16 channels. This token is not stored in the Flash Information Area. It is updated by the stack each time a calibration is performed.
EZSP_MFG_CBKE_DATA	0x09	Certificate Based Key Exchange (CBKE) data (92 bytes).
EZSP_MFG_INSTALLATION_CODE	0x0A	Installation code (20 bytes).
EZSP_STACK_CAL_FILTER	0x0B	Radio channel filter calibration data (1 byte). This token is not stored in the Flash Information Area. It is updated by the stack each time a calibration is performed.
EZSP_MFG_CUSTOM_EUI_64	0x0C	Custom EUI64 MAC address (8 bytes).
EZSP_MFG_CTUNE	0x0D	CTUNE value (2 byte).

EzspStatus	Value	Description
EZSP_SUCCESS	0x00	Success.
EZSP_SPI_ERR_FATAL	0x10	Fatal error.
EZSP_SPI_ERR_NCP_RESET	0x11	The Response frame of the current transaction indicates the NCP has reset.
EZSP_SPI_ERR_OVERSIZED_EZSP_FRAME	0x12	The NCP is reporting that the Command frame of the current transaction is oversized (the length byte is too large).
EZSP_SPI_ERR_ABORTED_TRANSACTION	0x13	The Response frame of the current transaction indicates the previous transaction was aborted (nSSEL deasserted too soon).
EZSP_SPI_ERR_MISSING_FRAME_TERMINATOR	0x14	The Response frame of the current transaction indicates the frame terminator is missing from the Command frame.
EZSP_SPI_ERR_WAIT_SECTION_TIMEOUT	0x15	The NCP has not provided a Response within the time limit defined by WAIT_SECTION_TIMEOUT.
EZSP_SPI_ERR_NO_FRAME_TERMINATOR	0x16	The Response frame from the NCP is missing the frame terminator.
EZSP_SPI_ERR_EZSP_COMMAND_OVERSIZED	0x17	The Host attempted to send an oversized Command (the length byte is too large) and the AVR's spi-protocol.c blocked the transmission.
EZSP_SPI_ERR_EZSP_RESPONSE_OVERSIZED	0x18	The NCP attempted to send an oversized Response (the length byte is too large) and the AVR's spi-protocol.c blocked the reception.
EZSP_SPI_WAITING_FOR_RESPONSE	0x19	The Host has sent the Command and is still waiting for the NCP to send a Response.
EZSP_SPI_ERR_HANDSHAKE_TIMEOUT	0x1A	The NCP has not asserted nHOST_INT within the time limit defined by WAKE_HANDSHAKE_TIMEOUT.
EZSP_SPI_ERR_STARTUP_TIMEOUT	0x1B	The NCP has not asserted nHOST_INT after an NCP reset within the time limit defined by STARTUP_TIMEOUT.
EZSP_SPI_ERR_STARTUP_FAIL	0x1C	The Host attempted to verify the SPI Protocol activity and version number, and the verification failed.
EZSP_SPI_ERR_UNSUPPORTED_SPI_COMMAND	0x1D	The Host has sent a command with a SPI Byte that is unsupported by the current mode the NCP is operating in.
EZSP_ASH_IN_PROGRESS	0x20	Operation not yet complete.
EZSP_HOST_FATAL_ERROR	0x21	Fatal error detected by host.
EZSP_ASH_NCP_FATAL_ERROR	0x22	Fatal error detected by NCP.
EZSP_DATA_FRAME_TOO_LONG	0x23	Tried to send DATA frame too long.
EZSP_DATA_FRAME_TOO_SHORT	0x24	Tried to send DATA frame too short.
EZSP_NO_TX_SPACE	0x25	No space for tx'ed DATA frame.
EZSP_NO_RX_SPACE	0x26	No space for rec'd DATA frame.
EZSP_NO_RX_DATA	0x27	No receive data available.
EZSP_NOT_CONNECTED	0x28	Not in Connected state.
EZSP_ERROR_VERSION_NOT_SET	0x30	The NCP received a command before the EZSP version had been set.
EZSP_ERROR_INVALID_FRAME_ID	0x31	The NCP received a command containing an unsupported frame ID.
EZSP_ERROR_WRONG_DIRECTION	0x32	The direction flag in the frame control field was incorrect.
EZSP_ERROR_TRUNCATED	0x33	The truncated flag in the frame control field was set, indicating there was not enough memory available to complete the response or that the response would have exceeded the maximum EZSP frame length.

EzspStatus	Value	Description
EZSP_ERROR_OVERFLOW	0x34	The overflow flag in the frame control field was set, indicating one or more callbacks occurred since the previous response and there was not enough memory available to report them to the Host.
EZSP_ERROR_OUT_OF_MEMORY	0x35	Insufficient memory was available.
EZSP_ERROR_INVALID_VALUE	0x36	The value was out of bounds.
EZSP_ERROR_INVALID_ID	0x37	The configuration id was not recognized.
EZSP_ERROR_INVALID_CALL	0x38	Configuration values can no longer be modified.
EZSP_ERROR_NO_RESPONSE	0x39	The NCP failed to respond to a command.
EZSP_ERROR_COMMAND_TOO_LONG	0x40	The length of the command exceeded the maximum EZSP frame length.
EZSP_ERROR_QUEUE_FULL	0x41	The UART receive queue was full causing a callback response to be dropped.
EZSP_ERROR_COMMAND_FILTERED	0x42	The command has been filtered out by NCP.
EZSP_ERROR_SECURITY_KEY_ALREADY_SET	0x43	EZSP Security Key is already set
EZSP_ERROR_SECURITY_TYPE_INVALID	0x44	EZSP Security Type is invalid
EZSP_ERROR_SECURITY_PARAMETERS_INVALID	0x45	EZSP Security Parameters are invalid
EZSP_ERROR_SECURITY_PARAMETERS_ALREADY_SET	0x46	EZSP Security Parameters are already set
EZSP_ERROR_SECURITY_KEY_NOT_SET	0x47	EZSP Security Key is not set
EZSP_ERROR_SECURITY_PARAMETERS_NOT_SET	0x48	EZSP Security Parameters are not set
EZSP_ERROR_UNSUPPORTED_CONTROL	0x49	Received frame with unsupported control byte
EZSP_ERROR_UNSECURE_FRAME	0x4A	Received frame is unsecure, when security is established
EZSP_ASH_ERROR_VERSION	0x50	Incompatible ASH version
EZSP_ASH_ERROR_TIMEOUTS	0x51	Exceeded max ACK timeouts
EZSP_ASH_ERROR_RESET_FAIL	0x52	Timed out waiting for RSTACK
EZSP_ASH_ERROR_NCP_RESET	0x53	Unexpected ncp reset
EZSP_ERROR_SERIAL_INIT	0x54	Serial port initialization failed
EZSP_ASH_ERROR_NCP_TYPE	0x55	Invalid ncp processor type
EZSP_ASH_ERROR_RESET_METHOD	0x56	Invalid ncp reset method
EZSP_ASH_ERROR_XON_XOFF	0x57	XON/XOFF not supported by host driver
EZSP_ASH_STARTED	0x70	ASH protocol started
EZSP_ASH_CONNECTED	0x71	ASH protocol connected
EZSP_ASH_DISCONNECTED	0x72	ASH protocol disconnected
EZSP_ASH_ACK_TIMEOUT	0x73	Timer expired waiting for ack
EZSP_ASH_CANCELLED	0x74	Frame in progress cancelled
EZSP_ASH_OUT_OF_SEQUENCE	0x75	Received frame out of sequence
EZSP_ASH_BAD_CRC	0x76	Received frame with CRC error
EZSP_ASH_COMM_ERROR	0x77	Received frame with comm error
EZSP_ASH_BAD_ACKNUM	0x78	Received frame with bad ackNum
EZSP_ASH_TOO_SHORT	0x79	Received frame shorter than minimum
EZSP_ASH_TOO_LONG	0x7A	Received frame longer than maximum
EZSP_ASH_BAD_CONTROL	0x7B	Received frame with illegal control byte

EzspStatus	Value	Description
EZSP_ASH_BAD_LENGTH	0x7C	Received frame with illegal length for its type
EZSP_ASH_ACK_RECEIVED	0x7D	Received ASH Ack
EZSP_ASH_ACK_SENT	0x7E	Sent ASH Ack
EZSP_ASH_NAK_RECEIVED	0x7F	Received ASH Nak
EZSP_ASH_NAK_SENT	0x80	Sent ASH Nak
EZSP_ASH_RST_RECEIVED	0x81	Received ASH RST
EZSP_ASH_RST_SENT	0x82	Sent ASH RST
EZSP_ASH_STATUS	0x83	ASH Status
EZSP_ASH_TX	0x84	ASH TX
EZSP_ASH_RX	0x85	ASH RX
EZSP_CPC_ERROR_INIT	0x86	Failed to connect to CPC daemon or failed to open CPC endpoint
EZSP_NO_ERROR	0xFF	No reset or error

EmberStatus	Value	Description
EMBER_SUCCESS	0x00	The generic 'no error' message.
EMBER_ERR_FATAL	0x01	The generic 'fatal error' message.
EMBER_BAD_ARGUMENT	0x02	An invalid value was passed as an argument to a function
EMBER_EEPROM_MFG_STACK_VERSION_MISMATCH	0x04	The manufacturing and stack token format in non-volatile memory is different than what the stack expects (returned at initialization).
EMBER_EEPROM_MFG_VERSION_MISMATCH	0x06	The manufacturing token format in non-volatile memory is different than what the stack expects (returned at initialization).
EMBER_EEPROM_STACK_VERSION_MISMATCH	0x07	The stack token format in non-volatile memory is different than what the stack expects (returned at initialization).
EMBER_NO_BUFFERS	0x18	There are no more buffers.
EMBER_SERIAL_INVALID_BAUD_RATE	0x20	Specified an invalid baud rate.
EMBER_SERIAL_INVALID_PORT	0x21	Specified an invalid serial port.
EMBER_SERIAL_TX_OVERFLOW	0x22	Tried to send too much data.
EMBER_SERIAL_RX_OVERFLOW	0x23	There was not enough space to store a received character and the character was dropped.
EMBER_SERIAL_RX_FRAME_ERROR	0x24	Detected a UART framing error.
EMBER_SERIAL_RX_PARITY_ERROR	0x25	Detected a UART parity error.
EMBER_SERIAL_RX_EMPTY	0x26	There is no received data to process.
EMBER_SERIAL_RX_OVERRUN_ERROR	0x27	The receive interrupt was not handled in time, and a character was dropped.
EMBER_MAC_TRANSMIT_QUEUE_FULL	0x39	The MAC transmit queue is full.
EMBER_MAC_UNKNOWN_HEADER_TYPE	0x3A	MAC header FCR error on receive.
EMBER_MAC_SCANNING	0x3D	The MAC can't complete this task because it is scanning.
EMBER_MAC_NO_DATA	0x31	No pending data exists for device doing a data poll.
EMBER_MAC_JOINED_NETWORK	0x32	Attempt to scan when we are joined to a network.
EMBER_MAC_BAD_SCAN_DURATION	0x33	Scan duration must be 0 to 14 inclusive. Attempt was made to scan with an incorrect duration value.

EmberStatus	Value	Description
EMBER_MAC_INCORRECT_SCAN_TYPE	0x34	emberStartScan was called with an incorrect scan type.
EMBER_MAC_INVALID_CHANNEL_MASK	0x35	emberStartScan was called with an invalid channel mask.
EMBER_MAC_COMMAND_TRANSMIT_FAILURE	0x36	Failed to scan current channel because we were unable to transmit the relevant MAC command.
EMBER_MAC_NO_ACK_RECEIVED	0x40	We expected to receive an ACK following the transmission, but the MAC level ACK was never received.
EMBER_MAC_INDIRECT_TIMEOUT	0x42	Indirect data message timed out before polled.
EMBER_SIM_EEPROM_ERASE_PAGE_GREEN	0x43	The Simulated EEPROM is telling the application that there is at least one flash page to be erased. The GREEN status means the current page has not filled above the ERASE_CRITICAL_THRESHOLD. The application should call the function halSimEepromErasePage when it can to erase a page.
EMBER_SIM_EEPROM_ERASE_PAGE_RED	0x44	The Simulated EEPROM is telling the application that there is at least one flash page to be erased. The RED status means the current page has filled above the ERASE_CRITICAL_THRESHOLD. Due to the shrinking availability of write space, there is a danger of data loss. The application must call the function halSimEepromErasePage as soon as possible to erase a page.
EMBER_SIM_EEPROM_FULL	0x45	The Simulated EEPROM has run out of room to write any new data and the data trying to be set has been lost. This error code is the result of ignoring the SIM_EEPROM_ERASE_PAGE_RED error code. The application must call the function halSimEepromErasePage to make room for any further calls to set a token.
EMBER_ERR_FLASH_WRITE_INHIBITED	0x46	A fatal error has occurred while trying to write data to the Flash. The target memory attempting to be programmed is already programmed. The flash write routines were asked to flip a bit from a 0 to 1, which is physically impossible and the write was therefore inhibited. The data in the flash cannot be trusted after this error.
EMBER_ERR_FLASH_VERIFY_FAILED	0x47	A fatal error has occurred while trying to write data to the Flash and the write verification has failed. The data in the flash cannot be trusted after this error, and it is possible this error is the result of exceeding the life cycles of the flash.
EMBER_SIM_EEPROM_INIT_1_FAILED	0x48	Attempt 1 to initialize the Simulated EEPROM has failed. This failure means the information already stored in Flash (or a lack thereof), is fatally incompatible with the token information compiled into the code image being run.
EMBER_SIM_EEPROM_INIT_2_FAILED	0x49	Attempt 2 to initialize the Simulated EEPROM has failed. This failure means Attempt 1 failed, and the token system failed to properly reload default tokens and reset the Simulated EEPROM.
EMBER_SIM_EEPROM_INIT_3_FAILED	0x4A	Attempt 3 to initialize the Simulated EEPROM has failed. This failure means one or both of the tokens TOKEN_MFG_NVDATA_VERSION or TOKEN_STACK_NVDATA_VERSION were incorrect

EmberStatus	Value	Description
		and the token system failed to properly reload default tokens and reset the Simulated EEPROM.
EMBER_ERR_FLASH_PROG_FAIL	0x4B	A fatal error has occurred while trying to write data to the flash, possibly due to write protection or an invalid address. The data in the flash cannot be trusted after this error, and it is possible this error is the result of exceeding the life cycles of the flash.
EMBER_ERR_FLASH_ERASE_FAIL	0x4C	A fatal error has occurred while trying to erase flash, possibly due to write protection. The data in the flash cannot be trusted after this error, and it is possible this error is the result of exceeding the life cycles of the flash.
EMBER_ERR_BOOTLOADER_TRAP_TABLE_BAD	0x58	The bootloader received an invalid message (failed attempt to go into bootloader).
EMBER_ERR_BOOTLOADER_TRAP_UNKNOWN	0x59	Bootloader received an invalid message (failed attempt to go into bootloader).
EMBER_ERR_BOOTLOADER_NO_IMAGE	0x5A	The bootloader cannot complete the bootload operation because either an image was not found or the image exceeded memory bounds.
EMBER_DELIVERY_FAILED	0x66	The APS layer attempted to send or deliver a message, but it failed.
EMBER_BINDING_INDEX_OUT_OF_RANGE	0x69	This binding index is out of range of the current binding table.
EMBER_ADDRESS_TABLE_INDEX_OUT_OF_RANGE	0x6A	This address table index is out of range for the current address table.
EMBER_INVALID_BINDING_INDEX	0x6C	An invalid binding table index was given to a function.
EMBER_INVALID_CALL	0x70	The API call is not allowed given the current state of the stack.
EMBER_COST_NOT_KNOWN	0x71	The link cost to a node is not known.
EMBER_MAX_MESSAGE_LIMIT_REACHED	0x72	The maximum number of in-flight messages (i.e. EMBER_APS_UNICAST_MESSAGE_COUNT) has been reached.
EMBER_MESSAGE_TOO_LONG	0x74	The message to be transmitted is too big to fit into a single over-the-air packet.
EMBER_BINDING_IS_ACTIVE	0x75	The application is trying to delete or overwrite a binding that is in use.
EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE	0x76	The application is trying to overwrite an address table entry that is in use.
EMBER_ADC_CONVERSION_DONE	0x80	Conversion is complete.
EMBER_ADC_CONVERSION_BUSY	0x81	Conversion cannot be done because a request is being processed.
EMBER_ADC_CONVERSION_DEFERRED	0x82	Conversion is deferred until the current request has been processed.
EMBER_ADC_NO_CONVERSION_PENDING	0x84	No results are pending.
EMBER_SLEEP_INTERRUPTED	0x85	Sleeping (for a duration) has been abnormally interrupted and exited prematurely.
EMBER_PHY_TX_UNDERFLOW	0x88	The transmit hardware buffer underflowed.
EMBER_PHY_TX_INCOMPLETE	0x89	The transmit hardware did not finish transmitting a packet.
EMBER_PHY_INVALID_CHANNEL	0x8A	An unsupported channel setting was specified.
EMBER_PHY_INVALID_POWER	0x8B	An unsupported power setting was specified.

EmberStatus	Value	Description
EMBER_PHY_TX_BUSY	0x8C	The packet cannot be transmitted because the physical MAC layer is currently transmitting a packet. (This is used for the MAC backoff algorithm.)
EMBER_PHY_TX_CCA_FAIL	0x8D	The transmit attempt failed because all CCA attempts indicated that the channel was busy
EMBER_PHY_OSCILLATOR_CHECK_FAILED	0x8E	The software installed on the hardware doesn't recognize the hardware radio type.
EMBER_PHY_ACK_RECEIVED	0x8F	The expected ACK was received after the last transmission.
EMBER_NETWORK_UP	0x90	The stack software has completed initialization and is ready to send and receive packets over the air.
EMBER_NETWORK_DOWN	0x91	The network is not operating.
EMBER_JOIN_FAILED	0x94	An attempt to join a network failed.
EMBER_MOVE_FAILED	0x96	After moving, a mobile node's attempt to re-establish contact with the network failed.
EMBER_CANNOT_JOIN_AS_ROUTER	0x98	An attempt to join as a router failed due to a ZigBee versus ZigBee Pro incompatibility. ZigBee devices joining ZigBee Pro networks (or vice versa) must join as End Devices, not Routers.
EMBER_NODE_ID_CHANGED	0x99	The local node ID has changed. The application can obtain the new node ID by calling emberGetNodeId().
EMBER_PAN_ID_CHANGED	0x9A	The local PAN ID has changed. The application can obtain the new PAN ID by calling emberGetPanId().
EMBER_NETWORK_OPENED	0x9C	The network has been opened for joining.
EMBER_NETWORK_CLOSED	0x9D	The network has been closed for joining.
EMBER_NO_BEACONS	0xAB	An attempt to join or rejoin the network failed because no router beacons could be heard by the joining node.
EMBER_RECEIVED_KEY_IN_THE_CLEAR	0xAC	An attempt was made to join a Secured Network using a pre-configured key, but the Trust Center sent back a Network Key in-the-clear when an encrypted Network Key was required.
EMBER_NO_NETWORK_KEY_RECEIVED	0xAD	An attempt was made to join a Secured Network, but the device did not receive a Network Key.
EMBER_NO_LINK_KEY_RECEIVED	0xAE	After a device joined a Secured Network, a Link Key was requested but no response was ever received.
EMBER_PRECONFIGURED_KEY_REQUIRED	0xAF	An attempt was made to join a Secured Network without a pre-configured key, but the Trust Center sent encrypted data using a pre-configured key.
EMBER_NOT_JOINED	0x93	The node has not joined a network.
EMBER_INVALID_SECURITY_LEVEL	0x95	The chosen security level (the value of EMBER_SECURITY_LEVEL) is not supported by the stack.
EMBER_NETWORK_BUSY	0xA1	A message cannot be sent because the network is currently overloaded.
EMBER_INVALID_ENDPOINT	0xA3	The application tried to send a message using an endpoint that it has not defined.
EMBER_BINDING_HAS_CHANGED	0xA4	The application tried to use a binding that has been remotely modified and the change has not yet been reported to the application.

EmberStatus	Value	Description
EMBER_INSUFFICIENT_RANDOM_DATA	0xA5	An attempt to generate random bytes failed because of insufficient random data from the radio.
EMBER_APS_ENCRYPTION_ERROR	0xA6	There was an error in trying to encrypt at the APS Level. This could result from either an inability to determine the long address of the recipient from the short address (no entry in the binding table) or there is no link key entry in the table associated with the destination, or there was a failure to load the correct key into the encryption core.
EMBER_SECURITY_STATE_NOT_SET	0xA8	There was an attempt to form or join a network with security without calling emberSetInitialSecurityState() first.
EMBER_KEY_TABLE_INVALID_ADDRESS	0xB3	There was an attempt to set an entry in the key table using an invalid long address. An entry cannot be set using either the local device's or Trust Center's IEEE address. Or an entry already exists in the table with the same IEEE address. An Address of all zeros or all F's are not valid addresses in 802.15.4.
EMBER_SECURITY_CONFIGURATION_INVALID	0xB7	There was an attempt to set a security configuration that is not valid given the other security settings.
EMBER_TOO_SOON_FOR_SWITCH_KEY	0xB8	There was an attempt to broadcast a key switch too quickly after broadcasting the next network key. The Trust Center must wait at least a period equal to the broadcast timeout so that all routers have a chance to receive the broadcast of the new network key.
EMBER_KEY_NOT_AUTHORIZED	0xBB	The message could not be sent because the link key corresponding to the destination is not authorized for use in APS data messages. APS Commands (sent by the stack) are allowed. To use it for encryption of APS data messages it must be authorized using a key agreement protocol (such as CBKE).
EMBER_SECURITY_DATA_INVALID	0xBD	The security data provided was not valid, or an integrity check failed.
EMBER_SOURCE_ROUTE_FAILURE	0xA9	A ZigBee route error command frame was received indicating that a source routed message from this node failed en route.
EMBER_MANY_TO_ONE_ROUTE_FAILURE	0xAA	A ZigBee route error command frame was received indicating that a message sent to this node along a many-to-one route failed en route. The route error frame was delivered by an ad-hoc search for a functioning route.
EMBER_STACK_AND_HARDWARE_MISMATCH	0xB0	A critical and fatal error indicating that the version of the stack trying to run does not match with the chip it is running on. The software (stack) on the chip must be replaced with software that is compatible with the chip.
EMBER_INDEX_OUT_OF_RANGE	0xB1	An index was passed into the function that was larger than the valid range.
EMBER_TABLE_FULL	0xB4	There are no empty entries left in the table.
EMBER_TABLE_ENTRY_ERASED	0xB6	The requested table entry has been erased and contains no valid data.
EMBER_LIBRARY_NOT_PRESENT	0xB5	The requested function cannot be executed because the library that contains the necessary functionality is not present.

EmberStatus	Value	Description
EMBER_OPERATION_IN_PROGRESS	0xBA	The stack accepted the command and is currently processing the request. The results will be returned via an appropriate handler.
EMBER_APPLICATION_ERROR_0	0xF0	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_1	0xF1	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_2	0xF2	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_3	0xF3	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_4	0xF4	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_5	0xF5	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_6	0xF6	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_7	0xF7	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_8	0xF8	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_9	0xF9	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_10	0xFA	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_11	0xFB	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_12	0xFC	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_13	0xFD	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_14	0xFE	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.
EMBER_APPLICATION_ERROR_15	0xFF	This error is reserved for customer application use. This will never be returned from any portion of the network stack or HAL.

EmberEventUnits	Value	Description
EMBER_EVENT_INACTIVE	0x00	The event is not scheduled to run.
EMBER_EVENT_MS_TIME	0x01	The execution time is in approximate milliseconds.
EMBER_EVENT_QS_TIME	0x02	The execution time is in 'binary' quarter seconds (256 approximate milliseconds each).
EMBER_EVENT_MINUTE_TIME	0x03	The execution time is in 'binary' minutes (65536 approximate milliseconds each).

EmberNodeType	Value	Description
EMBER_UNKNOWN_DEVICE	0x00	Device is not joined.
EMBER_COORDINATOR	0x01	Will relay messages and can act as a parent to other nodes.
EMBER_ROUTER	0x02	Will relay messages and can act as a parent to other nodes.
EMBER_END_DEVICE	0x03	Communicates only with its parent and will not relay messages.
EMBER_SLEEPY_END_DEVICE	0x04	An end device whose radio can be turned off to save power. The application must poll to receive messages.

EmberNetworkStatus	Value	Description
EMBER_NO_NETWORK	0x00	The node is not associated with a network in any way.
EMBER_JOINING_NETWORK	0x01	The node is currently attempting to join a network.
EMBER_JOINED_NETWORK	0x02	The node is joined to a network.
EMBER_JOINED_NETWORK_NO_PARENT	0x03	The node is an end device joined to a network but its parent is not responding.
EMBER_LEAVING_NETWORK	0x04	The node is in the process of leaving its current network.

EmberIncomingMessageType	Value	Description
EMBER_INCOMING_UNICAST	0x00	Unicast.
EMBER_INCOMING_UNICAST_REPLY	0x01	Unicast reply.
EMBER_INCOMING_MULTICAST	0x02	Multicast.
EMBER_INCOMING_MULTICAST_LOOPBACK	0x03	Multicast sent by the local device.
EMBER_INCOMING_BROADCAST	0x04	Broadcast.
EMBER_INCOMING_BROADCAST_LOOPBACK	0x05	Broadcast sent by the local device.
EMBER_INCOMING_MANY_TO_ONE_ROUTE_REQUEST	0x06	Many to one route request.

EmberOutgoingMessageType	Value	Description
EMBER_OUTGOING_DIRECT	0x00	Unicast sent directly to an EmberNodeId.
EMBER_OUTGOING_VIA_ADDRESS_TABLE	0x01	Unicast sent using an entry in the address table.
EMBER_OUTGOING_VIA_BINDING	0x02	Unicast sent using an entry in the binding table.
EMBER_OUTGOING_MULTICAST	0x03	Multicast message. This value is passed to emberMessageSentHandler() only. It may not be passed to emberSendUnicast().
EMBER_OUTGOING_BROADCAST	0x04	Broadcast message. This value is passed to emberMessageSentHandler() only. It may not be passed to emberSendUnicast().

EmberMacPassthroughType	Value	Description
EMBER_MAC_PASSTHROUGH_NONE	0x00	No MAC passthrough messages.
EMBER_MAC_PASSTHROUGH_SE_INTERPAN	0x01	SE InterPAN messages.
EMBER_MAC_PASSTHROUGH_EMBERNET	0x02	Legacy EmberNet messages.
EMBER_MAC_PASSTHROUGH_EMBERNET_SOURCE	0x04	Legacy EmberNet messages filtered by their source address.

EmberBindingType	Value	Description
EMBER_UNUSED_BINDING	0x00	A binding that is currently not in use.
EMBER_UNICAST_BINDING	0x01	A unicast binding whose 64-bit identifier is the destination EUI64.
EMBER_MANY_TO_ONE_BINDING	0x02	A unicast binding whose 64-bit identifier is the aggregator EUI64.
EMBER_MULTICAST_BINDING	0x03	A multicast binding whose 64-bit identifier is the group address. A multicast binding can be used to send messages to the group and to receive messages sent to the group.

EmberApsOption	Value	Description
EMBER_APS_OPTION_NONE	0x0000	No options.
EMBER_APS_OPTION_ENCRYPTION	0x0020	Send the message using APS Encryption, using the Link Key shared with the destination node to encrypt the data at the APS Level.
EMBER_APS_OPTION_RETRY	0x0040	Resend the message using the APS retry mechanism.
EMBER_APS_OPTION_ENABLE_ROUTE_DISCOVERY	0x0100	Causes a route discovery to be initiated if no route to the destination is known.
EMBER_APS_OPTION_FORCE_ROUTE_DISCOVERY	0x0200	Causes a route discovery to be initiated even if one is known.
EMBER_APS_OPTION_SOURCE_EUI64	0x0400	Include the source EUI64 in the network frame.
EMBER_APS_OPTION_DESTINATION_EUI64	0x0800	Include the destination EUI64 in the network frame.
EMBER_APS_OPTION_ENABLE_ADDRESS_DISCOVERY	0x1000	Send a ZDO request to discover the node ID of the destination, if it is not already know.
EMBER_APS_OPTION_POLL_RESPONSE	0x2000	Reserved.
EMBER_APS_OPTION_ZDO_RESPONSE_REQUIRED	0x4000	This incoming message is a ZDO request not handled by the EmberZNet stack, and the application is responsible for sending a ZDO response. This flag is used only when the ZDO is configured to have requests handled by the application. See the EZSP_CONFIG_APPLICATION_ZDO_FLAGS configuration parameter for more information.
EMBER_APS_OPTION_FRAGMENT	0x8000	This message is part of a fragmented message. This option may only be set for unicasts. The groupId field gives the index of this fragment in the low-order byte. If the low-order byte is zero this is the first fragment and the high-order byte contains the number of fragments in the message.

EzspNetworkScanType	Value	Description
EZSP_ENERGY_SCAN	0x00	An energy scan scans each channel for its RSSI value.
EZSP_ACTIVE_SCAN	0x01	An active scan scans each channel for available networks.

EmberJoinDecision	Value	Description
EMBER_USE_PRECONFIGURED_KEY	0x00	Allow the node to join. The joining node should have a pre-configured key. The security data sent to it will be encrypted with that key.
EMBER_SEND_KEY_IN_THE_CLEAR	0x01	Allow the node to join. Send the network key in-the-clear to the joining device.
EMBER_DENY_JOIN	0x02	Deny join.
EMBER_NO_ACTION	0x03	Take no action.

EmberInitialSecurityBitmask	Value	Description
EMBER_STANDARD_SECURITY_MODE	0x0000	This enables ZigBee Standard Security on the node.
EMBER_DISTRIBUTED_TRUST_CENTER_MODE	0x0002	This enables Distributed Trust Center Mode for the device forming the network. (Previously known as EMBER_NO_TRUST_CENTER_MODE)
EMBER_TRUST_CENTER_GLOBAL_LINK_KEY	0x0004	This enables a Global Link Key for the Trust Center. All nodes will share the same Trust Center Link Key.
EMBER_PRECONFIGURED_NETWORK_KEY_MODE	0x0008	This enables devices that perform MAC Association with a pre-configured Network Key to join the network. It is only set on the Trust Center.
EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY	0x0084	This denotes that the preconfiguredKey is not the actual Link Key but a Secret Key known only to the Trust Center. It is hashed with the IEEE Address of the destination device in order to create the actual Link Key used in encryption. This is bit is only used by the Trust Center. The joining device need not set this.
EMBER_HAVE_PRECONFIGURED_KEY	0x0100	This denotes that the preconfiguredKey element has valid data that should be used to configure the initial security state.
EMBER_HAVE_NETWORK_KEY	0x0200	This denotes that the networkKey element has valid data that should be used to configure the initial security state.
EMBER_GET_LINK_KEY_WHEN_JOINING	0x0400	This denotes to a joining node that it should attempt to acquire a Trust Center Link Key during joining. This is only necessary if the device does not have a pre-configured key.
EMBER_REQUIRE_ENCRYPTED_KEY	0x0800	This denotes that a joining device should only accept an encrypted network key from the Trust Center (using its pre-configured key). A key sent in-the-clear by the Trust Center will be rejected and the join will fail. This option is only valid when utilizing a pre-configured key.
EMBER_NO_FRAME_COUNTER_RESET	0x1000	This denotes whether the device should NOT reset its outgoing frame counters (both NWK and APS) when ::emberSetInitialSecurityState() is called. Normally it is advised to reset the frame counter before joining a new network. However in cases where a device is joining to the same network again (but not using ::emberRejoinNetwork()) it should keep the NWK and APS frame counters stored in its tokens.
EMBER_GET_PRECONFIGURED_KEY_FROM_INSTALL_CODE	0x2000	This denotes that the device should obtain its preconfigured key from an installation code stored in the manufacturing token. The token contains a value that will be hashed to obtain the actual preconfigured key. If that token is not valid, then the call to emberSetInitialSecurityState() will fail.

EmberInitialSecurityBitmask	Value	Description
EMBER_HAVE_TRUST_CENTER_EUI64	0x0040	This denotes that the ::EmberInitialSecurityState::preconfiguredTrustCenterEui64 has a value in it containing the trust center EUI64. The device will only join a network and accept commands from a trust center with that EUI64. Normally this bit is NOT set, and the EUI64 of the trust center is learned during the join process. When commissioning a device to join onto an existing network, which is using a trust center, and without sending any messages, this bit must be set and the field ::EmberInitialSecurityState::preconfiguredTrustCenterEui64 must be populated with the appropriate EUI64.

EmberCurrentSecurityBitmask	Value	Description
EMBER_STANDARD_SECURITY_MODE	0x0000	This denotes that the device is running in a network with ZigBee Standard Security.
EMBER_DISTRIBUTED_TRUST_CENTER_MODE	0x0002	This denotes that the device is running in a network without a centralized Trust Center.
EMBER_GLOBAL_LINK_KEY	0x0004	This denotes that the device has a Global Link Key. The Trust Center Link Key is the same across multiple nodes.
EMBER_HAVE_TRUST_CENTER_LINK_KEY	0x0010	This denotes that the node has a Trust Center Link Key.
EMBER_TRUST_CENTER_USES_HASHED_LINK_KEY	0x0084	This denotes that the Trust Center is using a Hashed Link Key.

EmberKeyType	Value	Description
EMBER_TRUST_CENTER_LINK_KEY	0x01	A shared key between the Trust Center and a device.
EMBER_CURRENT_NETWORK_KEY	0x03	The current active Network Key used by all devices in the network.
EMBER_NEXT_NETWORK_KEY	0x04	The alternate Network Key that was previously in use, or the newer key that will be switched to.
EMBER_APPLICATION_LINK_KEY	0x05	An Application Link Key shared with another (non-Trust Center) device.

EmberKeyStructBitmask	Value	Description
EMBER_KEY_HAS_SEQUENCE_NUMBER	0x0001	The key has a sequence number associated with it.
EMBER_KEY_HAS_OUTGOING_FRAME_COUNTER	0x0002	The key has an outgoing frame counter associated with it.
EMBER_KEY_HAS_INCOMING_FRAME_COUNTER	0x0004	The key has an incoming frame counter associated with it.
EMBER_KEY_HAS_PARTNER_EUI64	0x0008	The key has a Partner IEEE address associated with it.

EmberDeviceUpdate	Value
EMBER_STANDARD_SECURITY_SECURED_REJOIN	0x0
EMBER_STANDARD_SECURITY_UNSECURED_JOIN	0x1
EMBER_DEVICE_LEFT	0x2
EMBER_STANDARD_SECURITY_UNSECURED_REJOIN	0x3

EmberKeyStatus	Value
EMBER_APP_LINK_KEY_ESTABLISHED	0x01
EMBER_TRUST_CENTER_LINK_KEY_ESTABLISHED	0x03
EMBER_KEY_ESTABLISHMENT_TIMEOUT	0x04
EMBER_KEY_TABLE_FULL	0x05
EMBER_TC_RESPONDED_TO_KEY_REQUEST	0x06
EMBER_TC_APP_KEY_SENT_TO_REQUESTER	0x07
EMBER_TC_RESPONSE_TO_KEY_REQUEST_FAILED	0x08
EMBER_TC_REQUEST_KEY_TYPE_NOT_SUPPORTED	0x09
EMBER_TC_NO_LINK_KEY_FOR_REQUESTER	0x0A
EMBER_TC_REQUESTER_EUI64_UNKNOWN	0x0B
EMBER_TC_RECEIVED_FIRST_APP_KEY_REQUEST	0x0C
EMBER_TC_TIMEOUT_WAITING_FOR_SECOND_APP_KEY_REQUEST	0x0D
EMBER_TC_NON_MATCHING_APP_KEY_REQUEST_RECEIVED	0x0E
EMBER_TC_FAILED_TO_SEND_APP_KEYS	0x0F
EMBER_TC_FAILED_TO_STORE_APP_KEY_REQUEST	0x10
EMBER_TC_REJECTED_APP_KEY_REQUEST	0x11

EmberCounterType	Value	Description
EMBER_COUNTER_MAC_RX_BROADCAST	0	The MAC received a broadcast.
EMBER_COUNTER_MAC_TX_BROADCAST	1	The MAC transmitted a broadcast.
EMBER_COUNTER_MAC_RX_UNICAST	2	The MAC received a unicast.
EMBER_COUNTER_MAC_TX_UNICAST_SUCCESS	3	The MAC successfully transmitted a unicast.
EMBER_COUNTER_MAC_TX_UNICAST_RETRY	4	The MAC retried a unicast.
EMBER_COUNTER_MAC_TX_UNICAST_FAILED	5	The MAC unsuccessfully transmitted a unicast.
EMBER_COUNTER_APS_DATA_RX_BROADCAST	6	The APS layer received a data broadcast.
EMBER_COUNTER_APS_DATA_TX_BROADCAST	7	The APS layer transmitted a data broadcast.
EMBER_COUNTER_APS_DATA_RX_UNICAST	8	The APS layer received a data unicast.
EMBER_COUNTER_APS_DATA_TX_UNICAST_SUCCESS	9	The APS layer successfully transmitted a data unicast.
EMBER_COUNTER_APS_DATA_TX_UNICAST_RETRY	10	The APS layer retried a data unicast.
EMBER_COUNTER_APS_DATA_TX_UNICAST_FAILED	11	The APS layer unsuccessfully transmitted a data unicast.
EMBER_COUNTER_ROUTE_DISCOVERY_INITIATED	12	The network layer successfully submitted a new route discovery to the MAC.
EMBER_COUNTER_NEIGHBOR_ADDED	13	An entry was added to the neighbor table.
EMBER_COUNTER_NEIGHBOR_REMOVED	14	An entry was removed from the neighbor table.
EMBER_COUNTER_NEIGHBOR_STALE	15	A neighbor table entry became stale because it had not been heard from.
EMBER_COUNTER_JOIN_INDICATION	16	A node joined or rejoined to the network via this node.
EMBER_COUNTER_CHILD_REMOVED	17	An entry was removed from the child table.
EMBER_COUNTER_ASH_OVERFLOW_ERROR	18	EZSP-UART only. An overflow error occurred in the UART.
EMBER_COUNTER_ASH_FRAMING_ERROR	19	EZSP-UART only. A framing error occurred in the UART.

EmberCounterType	Value	Description
EMBER_COUNTER_ASH_OVERRUN_ERROR	20	EZSP-UART only. An overrun error occurred in the UART.
EMBER_COUNTER_NWK_FRAME_COUNTER_FAILURE	21	A message was dropped at the network layer because the NWK frame counter was not higher than the last message seen from that source.
EMBER_COUNTER_APS_FRAME_COUNTER_FAILURE	22	A message was dropped at the APS layer because the APS frame counter was not higher than the last message seen from that source.
EMBER_COUNTER_UTILITY	23	Utility counter for general debugging use.
EMBER_COUNTER_APS_LINK_KEY_NOT_AUTHORIZED	24	A message was dropped at the APS layer because it had APS encryption but the key associated with the sender has not been authenticated, and thus the key is not authorized for use in APS data messages.
EMBER_COUNTER_NWK_DECRYPTION_FAILURE	25	An NWK-encrypted message was received but dropped because decryption failed.
EMBER_COUNTER_APS_DECRYPTION_FAILURE	26	An APS encrypted message was received but dropped because decryption failed.
EMBER_COUNTER_ALLOCATE_PACKET_BUFFER_FAILURE	27	The number of times we failed to allocate a set of linked packet buffers. This doesn't necessarily mean that the packet buffer count was 0 at the time, but that the number requested was greater than the number free.
EMBER_COUNTER_RELAYED_UNICAST	28	The number of relayed unicast packets.
EMBER_COUNTER_PHY_TO_MAC_QUEUE_LIMIT_REACHED	29	The number of times we dropped a packet due to reaching the preset PHY to MAC queue limit (sli_802154mac_max_phy_to_mac_queue_length)
EMBER_COUNTER_PACKET_VALIDATE_LIBRARY_ DROPPED_COUNT	30	The number of times we dropped a packet due to the packet-validate library checking a packet and rejecting it due to length or other formatting problems.
EMBER_COUNTER_TYPE_NWK_RETRY_OVERFLOW	31	The number of times the NWK retry queue is full and a new message failed to be added.
EMBER_COUNTER_PHY_CCA_FAIL_COUNT	32	The number of times the PHY layer was unable to transmit due to a failed CCA.
EMBER_COUNTER_BROADCAST_TABLE_FULL	33	The number of times an NWK broadcast was dropped because the broadcast table was full.
EMBER_COUNTER_PTA_LO_PRI_REQUESTED	34	The number of low priority packet traffic arbitration requests.
EMBER_COUNTER_PTA_HI_PRI_REQUESTED	35	The number of high priority packet traffic arbitration requests.
EMBER_COUNTER_PTA_LO_PRI_DENIED	36	The number of low priority packet traffic arbitration requests denied.
EMBER_COUNTER_PTA_HI_PRI_DENIED	37	The number of high priority packet traffic arbitration requests denied.
EMBER_COUNTER_PTA_LO_PRI_TX_ABORTED	38	The number of aborted low priority packet traffic arbitration transmissions.
EMBER_COUNTER_PTA_HI_PRI_TX_ABORTED	39	The number of aborted high priority packet traffic arbitration transmissions.
EMBER_COUNTER_TYPE_COUNT	40	A placeholder giving the number of Ember counter types.

EmberJoinMethod	Value	Description
EMBER_USE_MAC_ASSOCIATION	0x0	Normally devices use MAC Association to join a network, which respects the "permit joining" flag in the MAC Beacon. This value should be used by default.
EMBER_USE_NWK_REJOIN	0x1	For those networks where the "permit joining" flag is never turned on, they will need to use a ZigBee NWK Rejoin. This value causes the rejoin to be sent without NWK security and the Trust Center will be asked to send the NWK key to the device. The NWK key sent to the device can be encrypted with the device's corresponding Trust Center link key. That is determined by the ::EmberJoinDecision on the Trust Center returned by the ::emberTrustCenterJoinHandler().
EMBER_USE_NWK_REJOIN_HAVE_NWK_KEY	0x2	For those networks where the "permit joining" flag is never turned on, they will need to use an NWK Rejoin. If those devices have been preconfigured with the NWK key (including sequence number) they can use a secured rejoin. This is only necessary for end devices since they need a parent. Routers can simply use the ::EMBER_USE_CONFIGURED_NWK_STATE join method below.
EMBER_USE_CONFIGURED_NWK_STATE	0x3	For those networks where all network and security information is known ahead of time, a router device may be commissioned such that it does not need to send any messages to begin communicating on the network.

EmberZdoConfigurationFlags	Value	Description
EMBER_APP_RECEIVES_SUPPORTED_ZDO_REQUESTS	0x01	Set this flag in order to receive supported ZDO request messages via the incomingMessageHandler callback. A supported ZDO request is one that is handled by the EmberZNet stack. The stack will continue to handle the request and send the appropriate ZDO response even if this configuration option is enabled.
EMBER_APP_HANDLES_UNSUPPORTED_ZDO_REQUESTS	0x02	Set this flag in order to receive unsupported ZDO request messages via the incomingMessageHandler callback. An unsupported ZDO request is one that is not handled by the EmberZNet stack, other than to send a 'not supported' ZDO response. If this configuration option is enabled, the stack will no longer send any ZDO response, and it is the application's responsibility to do so.
EMBER_APP_HANDLES_ZDO_ENDPOINT_REQUESTS	0x04	Set this flag in order to receive the following ZDO request messages via the incomingMessageHandler callback: SIMPLE_DESCRIPTOR_REQUEST, MATCH_DESCRIPTORS_REQUEST, and ACTIVE_ENDPOINTS_REQUEST. If this configuration option is enabled, the stack will no longer send any ZDO response for these requests, and it is the application's responsibility to do so.
EMBER_APP_HANDLES_ZDO_BINDING_REQUESTS	0x08	Set this flag in order to receive the following ZDO request messages via the incomingMessageHandler callback: BINDING_TABLE_REQUEST, BIND_REQUEST, and UNBIND_REQUEST. If this configuration option is enabled, the stack will no longer send any ZDO response for these requests, and it is the application's responsibility to do so.

EmberConcentratorType	Value	Description
EMBER_LOW_RAM_CONCENTRATOR	0xFFF8	A concentrator with insufficient memory to store source routes for the entire network. Route records are sent to the concentrator prior to every inbound APS unicast.
EMBER_HIGH_RAM_CONCENTRATOR	0xFFF9	A concentrator with sufficient memory to store source routes for the entire network. Remote nodes stop sending route records once the concentrator has successfully received one.

EmberZllState	Value	Description
EMBER_ZLL_STATE_NONE	0x0000	No state.
EMBER_ZLL_STATE_FACTORY_NEW	0x0001	The device is factory new.
EMBER_ZLL_STATE_ADDRESS_ASSIGNMENT_CAPABLE	0x0002	The device is capable of assigning addresses to other devices.
EMBER_ZLL_STATE_LINK_INITIATOR	0x0010	The device is initiating a link operation.
EMBER_ZLL_STATE_LINK_PRIORITY_REQUEST	0x0020	The device is requesting link priority.
EMBER_ZLL_STATE_NON_ZLL_NETWORK	0x0100	The device is on a non-ZLL network.

EmberZllKeyIndex	Value	Description
EMBER_ZLL_KEY_INDEX_DEVELOPMENT	0x00	Key encryption algorithm for use during development.
EMBER_ZLL_KEY_INDEX_MASTER	0x04	Key encryption algorithm shared by all certified devices.
EMBER_ZLL_KEY_INDEX_CERTIFICATION	0x0F	Key encryption algorithm for use during development and certification.

EzspZllNetworkOperation	Value	Description
EZSP_ZLL_FORM_NETWORK	0x00	ZLL form network command.
EZSP_ZLL_JOIN_TARGET	0x01	ZLL join target command.

EzspSourceRouteOverheadInformation	Value	Description
EZSP_SOURCE_ROUTE_OVERHEAD_UNKNOWN	0xFF	Ezsp source route overhead unknown

EmberNetworkInitBitmask	Value	Description
EMBER_NETWORK_INIT_NO_OPTIONS	0x0000	No options for Network Init
EMBER_NETWORK_INIT_PARENT_INFO_IN_TOKEN	0x0001	Save parent info (node ID and EUI64) in a token during joining/rejoin, and restore on reboot.
EMBER_NETWORK_INIT_END_DEVICE_REJOIN_ON_REBOOT	0x0002	Send a rejoin request as an end device on reboot if parent information is persisted.

EmberMultiPhyNwkConfig		
EMBER_BROADCAST_SUPPORT	0x01	Enable broadcast support on Routers

EmberDutyCycleState		
EMBER_DUTY_CYCLE_TRACKING_OFF	0	No Duty cycle tracking or metrics are taking place.
EMBER_DUTY_CYCLE_LBT_NORMAL	1	Duty Cycle is tracked and has not exceeded any thresholds.
EMBER_DUTY_CYCLE_LBT_LIMITED_THRESHOLD_REACHED	2	We have exceeded the limited threshold of our total duty cycle allotment.
EMBER_DUTY_CYCLE_LBT_CRITICAL_THRESHOLD_REACHED	3	We have exceeded the critical threshold of our total duty cycle allotment
EMBER_DUTY_CYCLE_LBT_SUSPEND_LIMIT_REACHED	4	We have reached the suspend limit and are blocking all outbound transmissions.

EmberRadioPowerMode		
EMBER_RADIO_POWER_MODE_RX_ON	0	The radio receiver is switched on.
EMBER_RADIO_POWER_MODE_OFF	1	The radio receiver is switched off.

EmberEntropySource		
EMBER_ENTROPY_SOURCE_ERROR	0	Entropy source error.
EMBER_ENTROPY_SOURCE_RADIO	1	Entropy source is the radio.
EMBER_ENTROPY_SOURCE_MBEDTLS_TRNG	2	Entropy source is the TRNG powered by mbed TLS.
EMBER_ENTROPY_SOURCE_MBEDTLS	3	Entropy source is powered by mbed TLS, the source is not TRNG.

sl_zb_sec_man_key_type_t		
SL_ZB_SEC_MAN_KEY_TYPE_NONE	0	No key type.
SL_ZB_SEC_MAN_KEY_TYPE_NETWORK	1	Network Key (either current or alternate).
SL_ZB_SEC_MAN_KEY_TYPE_TC_LINK	2	Preconfigured Trust Center Link Key.
SL_ZB_SEC_MAN_KEY_TYPE_TC_LINK_WITH_TIMEOUT	3	Transient key.
SL_ZB_SEC_MAN_KEY_TYPE_APP_LINK	4	Link key in table.
SL_ZB_SEC_MAN_KEY_TYPE_ZLL_ENCRYPTION_KEY	6	Encryption key in ZLL.
SL_ZB_SEC_MAN_KEY_TYPE_ZLL_PRECONFIGURED_KEY	7	Preconfigured key in ZLL.
SL_ZB_SEC_MAN_KEY_TYPE_GREEN_POWER_PROXY_TABLE_KEY	8	GP Proxy table key.
SL_ZB_SEC_MAN_KEY_TYPE_GREEN_POWER_SINK_TABLE_KEY	9	GP Sink table key.
SL_ZB_SEC_MAN_KEY_TYPE_INTERNAL	10	Generic key type available to use for crypto operations.

sl_zb_sec_man_derived_key_type_t		
SL_ZB_SEC_MAN_DERIVED_KEY_TYPE_NONE	0	No derivation (use core key type directly).
SL_ZB_SEC_MAN_DERIVED_KEY_TYPE_KEY_TRANSPORT_KEY	1	Hash core key with Key Transport Key hash.
SL_ZB_SEC_MAN_DERIVED_KEY_TYPE_KEY_LOAD_KEY	2	Hash core key with Key Load Key hash.
SL_ZB_SEC_MAN_DERIVED_KEY_TYPE_VERIFY_KEY	3	Perform Verify Key hash.
SL_ZB_SEC_MAN_DERIVED_KEY_TYPE_TC_SWAP_OUT_KEY	4	Perform a simple AES hash of the key for TC backup.
SL_ZB_SEC_MAN_DERIVED_KEY_TYPE_TC_HASHED_LINK_KEY	5	For a TC using hashed link keys, hashed the root key against the supplied EUI in context.

sl_zigbee_sec_man_flags_t		
ZB_SEC_MAN_FLAG_NONE	0	No flags on operation.
ZB_SEC_MAN_FLAG_KEY_INDEX_IS_VALID	1	Context has a valid key index.
ZB_SEC_MAN_FLAG_EUI_IS_VALID	2	Context has a valid EUI64.
ZB_SEC_MAN_FLAG_UNCONFIRMED_TRANSIENT_KEY	4	Transient key being added hasn't yet been verified.

4 Configuration Frames

Name: version	**ID:** 0x0000
Description: The command allows the Host to specify the desired EZSP version and must be sent before any other command. The response provides information about the firmware running on the NCP.	
Command Parameters:	
uint8_t desiredProtocolVersion	The EZSP version the Host wishes to use. To successfully set the version and allow other commands, this must be same as EZSP_PROTOCOL_VERSION.
Response Parameters:	
uint8_t protocolVersion	The EZSP version the NCP is using.
uint8_t stackType	The type of stack running on the NCP (2).
uint16_t stackVersion	The version number of the stack.

Name: getConfigurationValue	**ID:** 0x0052
Description: Reads a configuration value from the NCP.	
Command Parameters:	
EzspConfigId configId	Identifies which configuration value to read.
Response Parameters:	
EzspStatus status	EZSP_SUCCESS if the value was read successfully, EZSP_ERROR_INVALID_ID if the NCP does not recognize *configId*.
uint16_t value	The configuration value.

Name: setConfigurationValue	**ID:** 0x0053
Description: Writes a configuration value to the NCP. Configuration values can be modified by the Host after the NCP has reset. Once the status of the stack changes to EMBER_NETWORK_UP, configuration values can no longer be modified and this command will respond with EZSP_ERROR_INVALID_CALL.	
Command Parameters:	
EzspConfigId configId	Identifies which configuration value to change.
uint16_t value	The new configuration value.
Response Parameters:	
EzspStatus status	EZSP_SUCCESS if the configuration value was changed, EZSP_ERROR_OUT_OF_MEMORY if the new value exceeded the available memory, EZSP_ERROR_INVALID_VALUE if the new value was out of bounds, EZSP_ERROR_INVALID_ID if the NCP does not recognize *configId*, EZSP_ERROR_INVALID_CALL if configuration values can no longer be modified.

Name: readAttribute	**ID:** 0x0108
Description: Read attribute data on NCP endpoints.	
Command Parameters:	
uint8_t endpoint	Endpoint
uint16_t cluster	Cluster.
uint16_t attributeId	Attribute ID.
uint8_t mask	Mask.
uint16_t manufacturerCode	Manufacturer code.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
uint8_t dataType	Attribute data type.
uint8_t readLength	Length of attribute data.
uint8_t[] dataPtr	Attribute data.

Name: WriteAttribute	**ID:** 0x0109
Description: Write attribute data on NCP endpoints.	
Command Parameters:	
uint8_t endpoint	Endpoint
uint16_t cluster	Cluster.
uint16_t attributeId	Attribute ID.
uint8_t mask	Mask.
uint16_t manufacturerCode	Manufacturer code.
bool overrideReadOnlyAndDataType	Override read only and data type.
bool justTest	Override read only and data type.
uint8_t dataType	Attribute data type.
uint8_t dataLength	Attribute data length.
uint8_t[] data	Attribute data.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: addEndpoint	**ID:** 0x0002
Description: Configures endpoint information on the NCP. The NCP does not remember these settings after a reset. Endpoints can be added by the Host after the NCP has reset. Once the status of the stack changes to EMBER_NETWORK_UP, endpoints can no longer be added and this command will respond with EZSP_ERROR_INVALID_CALL.	
Command Parameters:	
uint8_t endpoint	The application endpoint to be added.
uint16_t profileId	The endpoint's application profile.
uint16_t deviceId	The endpoint's device ID within the application profile.
uint8_t appFlags	The device version and flags indicating description availability.
uint8_t inputClusterCount	The number of cluster IDs in *inputClusterList*.
uint8_t outputClusterCount	The number of cluster IDs in *outputClusterList*.
uint16_t[] inputClusterList	Input cluster IDs the endpoint will accept.
uint16_t[] outputClusterList	Output cluster IDs the endpoint may send.
Response Parameters:	
EzspStatus status	EZSP_SUCCESS if the endpoint was added, EZSP_ERROR_OUT_OF_MEMORY if there is not enough memory available to add the endpoint, EZSP_ERROR_INVALID_VALUE if the endpoint already exists, EZSP_ERROR_INVALID_CALL if endpoints can no longer be added.

Name: setPolicy	**ID:** 0x0055
Description: Allows the Host to change the policies used by the NCP to make fast decisions.	
Command Parameters:	
EzspPolicyId policyId	Identifies which policy to modify.
EzspDecisionId decisionId	The new decision for the specified policy.
Response Parameters:	
EzspStatus status	EZSP_SUCCESS if the policy was changed, EZSP_ERROR_INVALID_ID if the NCP does not recognize *policyId*.

Name: getPolicy	**ID:** 0x0056
Description: Allows the Host to read the policies used by the NCP to make fast decisions.	
Command Parameters:	
EzspPolicyId policyId	Identifies which policy to read.
Response Parameters:	
EzspStatus status	EZSP_SUCCESS if the policy was read successfully, EZSP_ERROR_INVALID_ID if the NCP does not recognize *policyId*.
EzspDecisionId decisionId	The current decision for the specified policy.

Name: sendPanIdUpdate	**ID:** 0x0057
Description: Triggers a pan id update message.	
Command Parameters:	
EmberPanId newPan	The new Pan Id
Response Parameters:	
bool status	true if the request was successfully handed to the stack, false otherwise

Name: getValue	**ID:** 0x00AA
Description: Reads a value from the NCP.	
Command Parameters:	
EzspValueId valueId	Identifies which value to read.
Response Parameters:	
EzspStatus status	EZSP_SUCCESS if the value was read successfully, EZSP_ERROR_INVALID_ID if the NCP does not recognize *valueId*, EZSP_ERROR_INVALID_VALUE if the length of the returned *value* exceeds the size of local storage allocated to receive it.
uint8_t valueLength	Both a command and response parameter. On command, the maximum size in bytes of local storage allocated to receive the returned *value*. On response, the actual length in bytes of the returned *value*.
uint8_t[] value	The value.

Name: getExtendedValue	**ID:** 0x0003
Description: Reads a value from the NCP but passes an extra argument specific to the value being retrieved.	
Command Parameters:	
EzspExtendedValueId valueId	Identifies which extended value ID to read.
uint32_t characteristics	Identifies which characteristics of the extended value ID to read. These are specific to the value being read.
Response Parameters:	
EzspStatus status	EZSP_SUCCESS if the value was read successfully, EZSP_ERROR_INVALID_ID if the NCP does not recognize *valueId*. EZSP_ERROR_INVALID_VALUE if the length of the returned *value* exceeds the size of local storage allocated to receive it.
uint8_t valueLength	Both a command and response parameter. On command, the maximum size in bytes of local storage allocated to receive the returned *value*. On response, the actual length in bytes of the returned *value*.
uint8_t[] value	The value.

Name: setValue	**ID:** 0x00AB
Description: Writes a value to the NCP.	
Command Parameters:	
EzspValueId valueId	Identifies which value to change.
uint8_t valueLength	The length of the *value* parameter in bytes.
uint8_t[] value	The new value.
Response Parameters:	
EzspStatus status	EZSP_SUCCESS if the value was changed, EZSP_ERROR_INVALID_VALUE if the new value was out of bounds, EZSP_ERROR_INVALID_ID if the NCP does not recognize *valueId*, EZSP_ERROR_INVALID_CALL if the value could not be modified.

Name: setPassiveAckConfig	**ID:** 0x0105
Description: Allows the Host to control the broadcast behaviour of a routing device used by the NCP	
Command Parameters:	
uint8_t config	Passive ack config enum.
uint8_t minAcksNeeded	The minimum number of acknowledgments (re-broadcasts) to wait for until deeming the broadcast transmission complete.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

5 Utilities Frames

Name: nop	**ID:** 0x0005
Description: A command which does nothing. The Host can use this to set the sleep mode or to check the status of the NCP.	
Command Parameters: None	
Response Parameters: None	

Name: echo	**ID:** 0x0081
Description: Variable length data from the Host is echoed back by the NCP. This command has no other effects and is designed for testing the link between the Host and NCP.	
Command Parameters:	
uint8_t dataLength	The length of the *data* parameter in bytes.
uint8_t[] data	The data to be echoed back.
Response Parameters:	
uint8_t echoLength	The length of the *echo* parameter in bytes.
uint8_t[] echo	The echo of the data.

Name: invalidCommand	**ID:** 0x0058
Description: Indicates that the NCP received an invalid command.	
This frame is a response to an invalid command.	
Response Parameters:	
EzspStatus reason	The reason why the command was invalid.

Name: callback	**ID:** 0x0006
Description: Allows the NCP to respond with a pending callback.	
Command Parameters: None	
The response to this command can be any of the callback responses.	

Name: noCallbacks	**ID:** 0x0007
Description: Indicates that there are currently no pending callbacks.	
This frame is a response to the *callback* command.	
Response Parameters: None	

Name: setToken	**ID:** 0x0009
Description: Sets a token (8 bytes of non-volatile storage) in the Simulated EEPROM of the NCP.	
Command Parameters:	
uint8_t tokenId	Which token to set.
uint8_t[8] tokenData	The data to write to the token.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: getToken	**ID:** 0x000A
Description: Retrieves a token (8 bytes of non-volatile storage) from the Simulated EEPROM of the NCP.	
Command Parameters:	
uint8_t tokenId	Which token to read.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
uint8_t[8] tokenData	The contents of the token.

Name: getMfgToken	**ID:** 0x000B	
Description: Retrieves a manufacturing token from the Flash Information Area of the NCP (except for EZSP_STACK_CAL_DATA which is managed by the stack).		
Command Parameters:		
EzspMfgTokenId tokenId	Which manufacturing token to read.	
Response Parameters:		
uint8_t tokenDataLength	The length of the *tokenData* parameter in bytes.	
uint8_t[] tokenData	The manufacturing token data.	

Name: setMfgToken	**ID:** 0x000C
Description: Sets a manufacturing token in the Customer Information Block (CIB) area of the NCP if that token currently unset (fully erased). Cannot be used with EZSP_STACK_CAL_DATA, EZSP_STACK_CAL_FILTER, EZSP_MFG_ASH_CONFIG, or EZSP_MFG_CBKE_DATA token.	
Command Parameters:	
EzspMfgTokenId tokenId	Which manufacturing token to set.
uint8_t tokenDataLength	The length of the *tokenData* parameter in bytes.
uint8_t[] tokenData	The manufacturing token data.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: stackTokenChangedHandler	**ID:** 0x000D
Description: A callback invoked to inform the application that a stack token has changed.	
This frame is a response to the *callback* command.	
Response Parameters:	
uint16_t tokenAddress	The address of the stack token that has changed.

Name: getRandomNumber	**ID:** 0x0049
Description: Returns a pseudorandom number.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	Always returns EMBER_SUCCESS.
uint16_t value	A pseudorandom number.

Name: setTimer	**ID:** 0x000E
Description: Sets a timer on the NCP. There are 2 independent timers available for use by the Host. A timer can be cancelled by setting *time* to 0 or *units* to EMBER_EVENT_INACTIVE.	
Command Parameters:	
uint8_t timerId	Which timer to set (0 or 1).
uint16_t time	The delay before the *timerHandler* callback will be generated. Note that the timer clock is free running and is not synchronized with this command. This means that the actual delay will be between *time* and (*time* - 1). The maximum delay is 32767.
EmberEventUnits units	The units for *time*.
bool repeat	If true, a *timerHandler* callback will be generated repeatedly. If false, only a single *timerHandler* callback will be generated.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: getTimer	**ID:** 0x004E
Description: Gets information about a timer. The Host can use this command to find out how much longer it will be before a previously set timer will generate a callback.	
Command Parameters:	
uint8_t timerId	Which timer to get information about (0 or 1).
Response Parameters:	
uint16_t time	The delay before the *timerHandler* callback will be generated.
EmberEventUnits units	The units for *time*.
bool repeat	True if a *timerHandler* callback will be generated repeatedly. False if only a single *timerHandler* callback will be generated.

Name: timerHandler	**ID:** 0x000F
Description: A callback from the timer.	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t timerId	Which timer generated the callback (0 or 1).

Name: debugWrite	**ID:** 0x0012
Description: Sends a debug message from the Host to the Network Analyzer utility via the NCP.	
Command Parameters:	
bool binaryMessage	true if the message should be interpreted as binary data, false if the message should be interpreted as ASCII text.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The binary message.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: readAndClearCounters	**ID:** 0x0065
Description: Retrieves and clears Ember counters. See the EmberCounterType enumeration for the counter types.	
Command Parameters: None	
Response Parameters:	
uint16_t[EMBER_COUNTER_TYPE_COUNT] values	A list of all counter values ordered according to the EmberCounterType enumeration.

Name: readCounters	**ID:** 0x00F1
Description: Retrieves Ember counters. See the EmberCounterType enumeration for the counter types.	
Command Parameters: None	
Response Parameters:	
uint16_t[EMBER_COUNTER_TYPE_COUNT] values	A list of all counter values ordered according to the EmberCounterType enumeration.

Name: counterRolloverHandler	**ID:** 0x00F2
Description: This call is fired when a counter exceeds its threshold.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberCounterType type	Type of Counter

Name: delayTest	**ID:** 0x009D
Description: Used to test that UART flow control is working correctly.	
Command Parameters:	
uint16_t delay	Data will not be read from the host for this many milliseconds.
Response Parameters: None	

Name: getLibraryStatus	**ID:** 0x0001
Description: This retrieves the status of the passed library ID to determine if it is compiled into the stack.	
Command Parameters:	
EmberLibraryId libraryId	The ID of the library being queried.
Response Parameters:	
EmberLibraryStatus status	The status of the library being queried.

Name: getXncpInfo	**ID:** 0x0013
Description: Allows the HOST to know whether the NCP is running the XNCP library. If so, the response contains also the manufacturer ID and the version number of the XNCP application that is running on the NCP.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if the NCP is running the XNCP library. EMBER_INVALID_CALL otherwise.
uint16_t manufacturerId	The manufactured ID the user has defined in the XNCP application.
uint16_t versionNumber	The version number of the XNCP application.

Name: customFrame	**ID:** 0x0047
Description: Provides the customer a custom EZSP frame. On the NCP, these frames are only handled if the XNCP library is included. On the NCP side these frames are handled in the emberXNcpIncomingCustomEzspMessageCallback() callback function.	
Command Parameters:	
uint8_t payloadLength	The length of the custom frame payload (maximum 119 bytes).
uint8_t[] payload	The payload of the custom frame.
Response Parameters:	
EmberStatus status	The status returned by the custom command.
uint8_t replyLength	The length of the response.
uint8_t[] reply	The response.

Name: customFrameHandler	**ID:** 0x0054
Description: A callback indicating a custom EZSP message has been received.	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t payloadLength	The length of the custom frame payload.
uint8_t[] payload	The payload of the custom frame.

Name: getEui64	**ID:** 0x0026
Description: Returns the EUI64 ID of the local node.	
Command Parameters: None	
Response Parameters:	
EmberEUI64 eui64	The 64-bit ID.

Name: getNodeId	**ID:** 0x0027
Description: Returns the 16-bit node ID of the local node.	
Command Parameters: None	
Response Parameters:	
EmberNodeId nodeId	The 16-bit ID.

Name: getPhyInterfaceCount	**ID:** 0x00FC
Description: Returns number of phy interfaces present.	
Command Parameters: None	
Response Parameters:	
uint8_t interfaceCount	Value indicate how many phy interfaces present.

Name: getTrueRandomEntropySource	**ID:** 0x004F
Description: Returns the entropy source used for true random number generation.	
Command Parameters: None	
Response Parameters:	
EmberEntropySource entropySource	Value indicates the used entropy source.

6 Networking Frames

Name: setManufacturerCode	**ID:** 0x0015
Description: Sets the manufacturer code to the specified value. The manufacturer code is one of the fields of the node descriptor.	
Command Parameters:	
uint16_t code	The manufacturer code for the local node.
Response Parameters: None	

Name: setPowerDescriptor	**ID:** 0x0016
Description: Sets the power descriptor to the specified value. The power descriptor is a dynamic value. Therefore, you should call this function whenever the value changes.	
Command Parameters:	
uint16_t descriptor	The new power descriptor for the local node.
Response Parameters: None	

Name: networkInit	**ID:** 0x0017
Description: Resume network operation after a reboot. The node retains its original type. This should be called on startup whether or not the node was previously part of a network. EMBER_NOT_JOINED is returned if the node is not part of a network. This command accepts options to control the network initialization.	
Command Parameters:	
EmberNetworkInitStruct networkInitStruct	An EmberNetworkInitStruct containing the options for initialization.
Response Parameters:	
EmberStatus status	An EmberStatus value that indicates one of the following: successful initialization, EMBER_NOT_JOINED if the node is not part of a network, or the reason for failure.

Name: networkState	**ID:** 0x0018
Description: Returns a value indicating whether the node is joining, joined to, or leaving a network.	
Command Parameters: None	
Response Parameters:	
EmberNetworkStatus status	An EmberNetworkStatus value indicating the current join status.

Name: stackStatusHandler	**ID:** 0x0019
Description: A callback invoked when the status of the stack changes. If the status parameter equals EMBER_NETWORK_UP, then the *getNetworkParameters* command can be called to obtain the new network parameters. If any of the parameters are being stored in nonvolatile memory by the Host, the stored values should be updated.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	Stack status.

Name: startScan	**ID:** 0x001A
Description: This function will start a scan.	
Command Parameters:	
EzspNetworkScanType scanType	Indicates the type of scan to be performed. Possible values are: EZSP_ENERGY_SCAN and EZSP_ACTIVE_SCAN. For each type, the respective callback for reporting results is: energyScanResultHandler and networkFoundHandler. The energy scan and active scan report errors and completion via the scanCompleteHandler.
uint32_t channelMask	Bits set as 1 indicate that this particular channel should be scanned. Bits set to 0 indicate that this particular channel should not be scanned. For example, a channelMask value of 0x00000001 would indicate that only channel 0 should be scanned. Valid channels range from 11 to 26 inclusive. This translates to a channel mask value of 0x07FFF800. As a convenience, a value of 0 is reinterpreted as the mask for the current channel.
uint8_t duration	Sets the exponent of the number of scan periods, where a scan period is 960 symbols. The scan will occur for ((2^duration) + 1) scan periods.
Response Parameters:	
sl_status_t status	SL_STATUS_OK signals that the scan successfully started. Possible error responses and their meanings: SL_STATUS_MAC_SCANNING, we are already scanning; SL_STATUS_BAD_SCAN_DURATION, we have set a duration value that is not 0..14 inclusive; SL_STATUS_MAC_INCORRECT_SCAN_TYPE, we have requested an undefined scanning type; SL_STATUS_INVALID_CHANNEL_MASK, our channel mask did not specify any valid channels.

Name: energyScanResultHandler	**ID:** 0x0048
Description: Reports the result of an energy scan for a single channel. The scan is not complete until the *scanCompleteHandler* callback is called.	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t channel	The 802.15.4 channel number that was scanned.
int8s maxRssiValue	The maximum RSSI value found on the channel.

Name: networkFoundHandler	**ID:** 0x001B
Description: Reports that a network was found as a result of a prior call to *startScan.* Gives the network parameters useful for deciding which network to join.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberZigbeeNetwork networkFound	The parameters associated with the network found.
uint8_t lastHopLqi	The link quality from the node that generated this beacon.
int8s lastHopRssi	The energy level (in units of dBm) observed during the reception.

Name: scanCompleteHandler	**ID:** 0x001C
Description: Returns the status of the current scan of type EZSP_ENERGY_SCAN or EZSP_ACTIVE_SCAN. EMBER_SUCCESS signals that the scan has completed. Other error conditions signify a failure to scan on the channel specified.	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t channel	The channel on which the current error occurred. Undefined for the case of EMBER_SUCCESS.
EmberStatus status	The error condition that occurred on the current channel. Value will be EMBER_SUCCESS when the scan has completed.

Name: unusedPanIdFoundHandler	**ID:** 0x00D2
Description: This function returns an unused panID and channel pair found via the find unused panId scan procedure.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberPanId panId	The unused panID which has been found.
uint8_t channel	The channel that the unused panID was found on.

Name: findUnusedPanId	**ID:** 0x00D3
Description: This function starts a series of scans which will return an available panId.	
Command Parameters:	
uint32_t channelMask	The channels that will be scanned for available panIds.
uint8_t duration	The duration of the procedure.
Response Parameters:	
EmberStatus status	The error condition that occurred during the scan. Value will be EMBER_SUCCESS if there are no errors.

Name: stopScan	**ID:** 0x001D
Description: Terminates a scan in progress.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: formNetwork	**ID:** 0x001E
Description: Forms a new network by becoming the coordinator.	
Command Parameters:	
EmberNetworkParameters parameters	Specification of the new network.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: joinNetwork	**ID:** 0x001F
Description: Causes the stack to associate with the network using the specified network parameters. It can take several seconds for the stack to associate with the local network. Do not send messages until the *stackStatusHandler* callback informs you that the stack is up.	
Command Parameters:	
EmberNodeType nodeType	Specification of the role that this node will have in the network. This role must not be EMBER_COORDINATOR. To be a coordinator, use the *formNetwork* command.
EmberNetworkParameters parameters	Specification of the network with which the node should associate.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: joinNetworkDirectly	**ID:** 0x003B
Description: Causes the stack to associate with the network using the specified network parameters in the beacon parameter. It can take several seconds for the stack to associate with the local network. Do not send messages until the *stackStatusHandler* callback informs you that the stack is up. Unlike ::emberJoinNetwork(), this function does not issue an active scan before joining. Instead, it will cause the local node to issue a MAC Association Request directly to the specified target node. It is assumed that the beacon parameter is an artifact after issuing an active scan. (For more information, see *emberGetBestBeacon* and *emberGetNextBeacon*.)	
Command Parameters:	
EmberNodeType localNodeType	Specifies the role that this node will have in the network. This role must not be EMBER_COORDINATOR. To be a coordinator, use the *formNetwork* command.
EmberBeaconData beacon	Specifies the network with which the node should associate.
int8_t radioTxPower	The radio transmit power to use, specified in dBm.
bool clearBeaconsAfterNetworkUp	If true, clear beacons in cache upon join success. If join fail, do nothing.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: leaveNetwork	**ID:** 0x0020
Description: Causes the stack to leave the current network. This generates a *stackStatusHandler* callback to indicate that the network is down. The radio will not be used until after sending a *formNetwork* or *joinNetwork* command.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: findAndRejoinNetwork	**ID:** 0x0021
Description: The application may call this function when contact with the network has been lost. The most common usage case is when an end device can no longer communicate with its parent and wishes to find a new one. Another case is when a device has missed a Network Key update and no longer has the current Network Key. The stack will call *ezspStackStatusHandler* to indicate that the network is down, then try to re-establish contact with the network by performing an active scan, choosing a network with matching extended pan id, and sending a ZigBee network rejoin request. A second call to the *ezspStackStatusHandler* callback indicates either the success or the failure of the attempt. The process takes approximately 150 milliseconds per channel to complete. This call replaces the *emberMobileNodeHasMoved* API from EmberZNet 2.x, which used MAC association and consequently took half a second longer to complete.	
Command Parameters:	
bool haveCurrentNetworkKey	This parameter tells the stack whether to try to use the current network key. If it has the current network key it will perform a secure rejoin (encrypted). If this fails the device should try an unsecure rejoin. If the Trust Center allows the rejoin then the current Network Key will be sent encrypted using the device's Link Key.
uint32_t channelMask	A mask indicating the channels to be scanned. See *emberStartScan* for format details. A value of 0 is reinterpreted as the mask for the current channel.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: permitJoining	**ID:** 0x0022
Description: Tells the stack to allow other nodes to join the network with this node as their parent. Joining is initially disabled by default.	
Command Parameters:	
uint8_t duration	A value of 0x00 disables joining. A value of 0xFF enables joining. Any other value enables joining for that number of seconds.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: childJoinHandler	**ID:** 0x0023
Description: Indicates that a child has joined or left.	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t index	The index of the child of interest.
bool joining	True if the child is joining. False the child is leaving.
EmberNodeId childId	The node ID of the child.
EmberEUI64 childEui64	The EUI64 of the child.
EmberNodeType childType	The node type of the child.

Name: energyScanRequest	**ID:** 0x009C
Description: Sends a ZDO energy scan request. This request may only be sent by the current network manager and must be unicast, not broadcast. See ezsp-utils.h for related macros emberSetNetworkManagerRequest() and emberChangeChannelRequest().	
Command Parameters:	
EmberNodeId target	The network address of the node to perform the scan.
uint32_t scanChannels	A mask of the channels to be scanned.
uint8_t scanDuration	How long to scan on each channel. Allowed values are 0..5, with the scan times as specified by 802.15.4 (0 = 31ms, 1 = 46ms, 2 = 77ms, 3 = 138ms, 4 = 261ms, 5 = 507ms).
uint16_t scanCount	The number of scans to be performed on each channel (1..8).
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: getNetworkParameters	**ID:** 0x0028
Description: Returns the current network parameters.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
EmberNodeType nodeType	An EmberNodeType value indicating the current node type.
EmberNetworkParameters parameters	The current network parameters.

Name: getRadioParameters	**ID:** 0x00FD
Description: Returns the current radio parameters based on phy index.	
Command Parameters:	
uint8_t phyIndex	Desired index of phy interface for radio parameters.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
EmberMultiPhyRadioParameters parameters	The current radio parameters based on provided phy index.

Name: getParentChildParameters	**ID:** 0x0029
Description: Returns information about the children of the local node and the parent of the local node.	
Command Parameters: None	
Response Parameters:	
uint8_t childCount	The number of children the node currently has.
EmberEUI64 parentEui64	The parent's EUI64. The value is undefined for nodes without parents (coordinators and nodes that are not joined to a network).
EmberNodeId parentNodeId	The parent's node ID. The value is undefined for nodes without parents (coordinators and nodes that are not joined to a network).

Name: getChildData	**ID:** 0x004A
Description: Returns information about a child of the local node.	
Command Parameters:	
uint8_t index	The index of the child of interest in the child table. Possible indexes range from zero to EMBER_CHILD_TABLE_SIZE.
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if there is a child at *index*. EMBER_NOT_JOINED if there is no child at *index*.
EmberChildData childData	The data of the child.

Name: setChildData	**ID:** 0x00AC
Description: Sets child data to the child table token.	
Command Parameters:	
uint8_t index	The index of the child of interest in the child table. Possible indexes range from zero to (EMBER_CHILD_TABLE_SIZE - 1).
EmberChildData childData	The data of the child.
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if the child data is set successfully at *index*. EMBER_INDEX_OUT_OF_RANGE if provided *index* is out of range.

Name: childId	**ID:** 0x0106
Description: Convert a child index to a node ID	
Command Parameters:	
uint8_t childIndex	The index of the child of interest in the child table. Possible indexes range from zero to EMBER_CHILD_TABLE_SIZE.
Response Parameters:	
EmberNodeId childId	The node ID of the child or EMBER_NULL_NODE_ID if there isn't a child at the childIndex specified

Name: ID	**ID:** 0x0107
Description: Convert a node ID to a child index	
Command Parameters:	
EmberNodeId childId	The node ID of the child
Response Parameters:	
uint8_t childIndex	The child index or 0xFF if the node ID doesn't belong to a child

Name: getSourceRouteTableTotalSize	**ID:** 0x00C3
Description: Returns the source route table total size.	
Command Parameters: None	
Response Parameters:	
uint8_t sourceRouteTableTotalSize	Total size of source route table.

Name: getSourceRouteTableFilledSize	**ID:** 0x00C2
Description: Returns the number of filled entries in source route table.	
Command Parameters: None	
Response Parameters:	
uint8_t sourceRouteTableFilledSize	The number of filled entries in source route table.

Name: getSourceRouteTableEntry	**ID:** 0x00C1
Description: Returns information about a source route table entry	
Command Parameters:	
uint8_t index	The index of the entry of interest in the source route table. Possible indexes range from zero to SOURCE_ROUTE_TABLE_FILLED_SIZE.
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if there is source route entry at *index*. EMBER_NOT_FOUND if there is no source route at *index*.
EmberNodeId destination	The node ID of the destination in that entry.
uint8_t closerIndex	The closer node index for this source route table entry

Name: getNeighbor	**ID:** 0x0079
Description: Returns the neighbor table entry at the given index. The number of active neighbors can be obtained using the neighborCount command.	
Command Parameters:	
uint8_t index	The index of the neighbor of interest. Neighbors are stored in ascending order by node id, with all unused entries at the end of the table.
Response Parameters:	
EmberStatus status	EMBER_ERR_FATAL if the index is greater or equal to the number of active neighbors, or if the device is an end device. Returns EMBER_SUCCESS otherwise.
EmberNeighborTableEntry value	The contents of the neighbor table entry.

Name: getNeighborFrameCounter	**ID:** 0x003E
Description: Return EmberStatus depending on whether the frame counter of the node is found in the neighbor or child table. This function gets the last received frame counter as found in the Network Auxiliary header for the specified neighbor or child	
Command Parameters:	
EmberEUI64 eui64	eui64 of the node
Response Parameters:	
EmberStatus status	Return EMBER_NOT_FOUND if the node is not found in the neighbor or child table. Returns EMBER_SUCCESS otherwise
uint32_t returnFrameCounter	Return the frame counter of the node from the neighbor or child table

Name: setNeighborFrameCounter	**ID:** 0x00AD
Description: Sets the frame counter for the neighbor or child.	
Command Parameters:	
EmberEUI64 eui64	eui64 of the node
uint32_t frameCounter	Return the frame counter of the node from the neighbor or child table
Response Parameters:	
EmberStatus status	Return EMBER_NOT_FOUND if the node is not found in the neighbor or child table. Returns EMBER_SUCCESS otherwise

Name: setRoutingShortcutThreshold	**ID:** 0x00D0
Description: Sets the routing shortcut threshold to directly use a neighbor instead of performing routing.	
Command Parameters:	
uint8_t costThresh	The routing shortcut threshold to configure.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: getRoutingShortcutThreshold	**ID:** 0x00D1
Description: Gets the routing shortcut threshold used to differentiate between directly using a neighbor vs. performing routing.	
Command Parameters: None	
Response Parameters:	
uint8_t routingShortcutThresh	The routing shortcut threshold

Name: neighborCount	**ID:** 0x007A
Description: Returns the number of active entries in the neighbor table.	
Command Parameters: None	
Response Parameters:	
uint8_t value	The number of active entries in the neighbor table.

Name: getRouteTableEntry	**ID:** 0x007B
Description: Returns the route table entry at the given index. The route table size can be obtained using the getConfigurationValue command.	
Command Parameters:	
uint8_t index	The index of the route table entry of interest.
Response Parameters:	
EmberStatus status	EMBER_ERR_FATAL if the index is out of range or the device is an end device, and EMBER_SUCCESS otherwise.
EmberRouteTableEntry value	The contents of the route table entry.

Name: setRadioPower	**ID:** 0x0099
Description: Sets the radio output power at which a node is operating. Ember radios have discrete power settings. For a list of available power settings, see the technical specification for the RF communication module in your Developer Kit. Note: Care should be taken when using this API on a running network, as it will directly impact the established link qualities neighboring nodes have with the node on which it is called. This can lead to disruption of existing routes and erratic network behavior.	
Command Parameters:	
int8s power	Desired radio output power, in dBm.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating the success or failure of the command.

Name: setRadioChannel	**ID:** 0x009A
Description: Sets the channel to use for sending and receiving messages. For a list of available radio channels, see the technical specification for the RF communication module in your Developer Kit. Note: Care should be taken when using this API, as all devices on a network must use the same channel.	
Command Parameters:	
uint8_t channel	Desired radio channel.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating the success or failure of the command.

Name: getRadioChannel	**ID:** 0x00FF
Description: Gets the channel in use for sending and receiving messages.	
Command Parameters: None	
Response Parameters:	
uint8_t channel	Current radio channel.

Name: setRadioIeee802154CcaMode	**ID:** 0x0095
Description: Set the configured 802.15.4 CCA mode in the radio.	
Command Parameters:	
uint8_t ccaMode	A RAIL_IEEE802154_CcaMode_t value.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating the success or failure of the command.

Name: setConcentrator	**ID:** 0x0010
Description: Enable/disable concentrator support.	
Command Parameters:	
bool on	If this bool is true the concentrator support is enabled. Otherwise is disabled. If this bool is false all the other arguments are ignored.
uint16_t concentratorType	Must be either EMBER_HIGH_RAM_CONCENTRATOR or EMBER_LOW_RAM_CONCENTRATOR. The former is used when the caller has enough memory to store source routes for the whole network. In that case, remote nodes stop sending route records once the concentrator has successfully received one. The latter is used when the concentrator has insufficient RAM to store all outbound source routes. In that case, route records are sent to the concentrator prior to every inbound APS unicast.
uint16_t minTime	The minimum amount of time that must pass between MTORR broadcasts.
uint16_t maxTime	The maximum amount of time that can pass between MTORR broadcasts.
uint8_t routeErrorThreshold	The number of route errors that will trigger a re-broadcast of the MTORR.
uint8_t deliveryFailureThreshold	The number of APS delivery failures that will trigger a re-broadcast of the MTORR.
uint8_t maxHops	The maximum number of hops that the MTORR broadcast will be allowed to have. A value of 0 will be converted to the EMBER_MAX_HOPS value set by the stack.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: setBrokenRouteErrorCode	**ID:** 0x0011
Description: Sets the error code that is sent back from a router with a broken route.	
Command Parameters:	
uint8_t errorCode	Desired error code.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating the success or failure of the command.

Name: multiPhyStart	**ID:** 0x00F8
Description: This causes to initialize the desired radio interface other than native and form a new network by becoming the coordinator with same panId as native radio network.	
Command Parameters:	
uint8_t phyIndex	Index of phy interface. The native phy index would be always zero hence valid phy index starts from one.
uint8_t page	Desired radio channel page.
uint8_t channel	Desired radio channel.
int8_t power	Desired radio output power, in dBm.
EmberMultiPhyNwkConfig bitmask	Network configuration bitmask.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: multiPhyStop	**ID:** 0x00F9
Description: This causes to bring down the radio interface other than native.	
Command Parameters:	
uint8_t phyIndex	Index of phy interface. The native phy index would be always zero hence valid phy index starts from one.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: multiPhySetRadioPower	**ID:** 0x00FA
Description: Sets the radio output power for desired phy interface at which a node is operating. Ember radios have discrete power settings. For a list of available power settings, see the technical specification for the RF communication module in your Developer Kit. Note: Care should be taken when using this api on a running network, as it will directly impact the established link qualities neighboring nodes have with the node on which it is called. This can lead to disruption of existing routes and erratic network behavior.	
Command Parameters:	
uint8_t phyIndex	Index of phy interface. The native phy index would be always zero hence valid phy index starts from one.
int8_t power	Desired radio output power, in dBm.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating the success or failure of the command.

Name: sendLinkPowerDeltaRequest	**ID:** 0x00F7
Description: Send Link Power Delta Request from a child to its parent	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating the success or failure of sending the request.

Name: multiPhySetRadioChannel	**ID:** 0x00FB
Description: Sets the channel for desired phy interface to use for sending and receiving messages. For a list of available radio pages and channels, see the technical specification for the RF communication module in your Developer Kit. Note: Care should be taken when using this API, as all devices on a network must use the same page and channel.	
Command Parameters:	
uint8_t phyIndex	Index of phy interface. The native phy index would be always zero hence valid phy index starts from one.
uint8_t page	Desired radio channel page.
uint8_t channel	Desired radio channel.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating the success or failure of the command.

Name: getDutyCycleState	**ID:** 0x0035
Description: Obtains the current duty cycle state.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating the success or failure of the command.
EmberDutyCycleState returnedState	The current duty cycle state in effect.

Name: setDutyCycleLimitsInStack	**ID:** 0x0040
Description: Set the current duty cycle limits configuration. The Default limits set by stack if this call is not made.	
Command Parameters:	
EmberDutyCycleLimits limits	The duty cycle limits configuration to utilize.
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if the duty cycle limit configurations set successfully, EMBER_BAD_ARGUMENT if set illegal value such as setting only one of the limits to default or violates constraints Susp > Crit > Limi, EMBER_INVALID_CALL if device is operating on 2.4Ghz

Name: getDutyCycleLimits	**ID:** 0x004B
Description: Obtains the current duty cycle limits that were previously set by a call to emberSetDutyCycleLimitsInStack(), or the defaults set by the stack if no set call was made.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating the success or failure of the command.
EmberDutyCycleLimits returnedLimits	Return current duty cycle limits if returnedLimits is not NULL

Name: getCurrentDutyCycle	**ID:** 0x004C
Description: Returns the duty cycle of the stack's connected children that are being monitored, up to maxDevices. It indicates the amount of overall duty cycle they have consumed (up to the suspend limit). The first entry is always the local stack's nodeId, and thus the total aggregate duty cycle for the device. The passed pointer arrayOfDeviceDutyCycles MUST have space for maxDevices.	
Command Parameters:	
uint8_t maxDevices	Number of devices to retrieve consumed duty cycle.
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if the duty cycles were read successfully, EMBER_BAD_ARGUMENT maxDevices is greater than EMBER_MAX_END_DEVICE_CHILDREN + 1.
uint8_t[134] arrayOfDeviceDutyCycles	Consumed duty cycles up to maxDevices. When the number of children that are being monitored is less than maxDevices, the EmberNodeId element in the EmberPerDeviceDutyCycle will be 0xFFFF.

Name: dutyCycleHandler	**ID:** 0x004D
Description: Callback fires when the duty cycle state has changed	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t channelPage	The channel page whose duty cycle state has changed.
uint8_t channel	The channel number whose duty cycle state has changed.
EmberDutyCycleState state	The current duty cycle state.
uint8_t totalDevices	The total number of connected end devices that are being monitored for duty cycle.
EmberPerDeviceDutyCycle arrayOf-DeviceDutyCycles	Consumed duty cycles of end devices that are being monitored. The first entry always be the local stack's nodeId, and thus the total aggregate duty cycle for the device.

Name: getFirstBeacon	**ID:** 0x003D
Description: Returns the first beacon in the cache. Beacons are stored in cache after issuing an active scan.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if first beacon found, EMBER_BAD_ARGUMENT if input parameters are invalid, EMBER_INVALID_CALL if no beacons stored, EMBER_ERR_FATAL if no first beacon found.
EmberBeaconIterator beaconIterator	The iterator to use when returning the first beacon. This argument must not be NULL.

Name: getNextBeacon	**ID:** 0x0004
Description: Returns the next beacon in the cache. Beacons are stored in cache after issuing an active scan.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if next beacon found, EMBER_BAD_ARGUMENT if input parameters are invalid, EMBER_ERR_FATAL if no next beacon found.
EmberBeaconData beacon	The next beacon retrieved. It is assumed that *emberGetFirstBeacon* has been called first. This argument must not be NULL.

Name: getNumStoredBeacons	**ID:** 0x0008
Description: Returns the number of cached beacons that have been collected from a scan.	
Command Parameters: None	
Response Parameters:	
uint8_t numBeacons	The number of cached beacons that have been collected from a scan.

Name: clearStoredBeacons	**ID:** 0x003C
Description: Clears all cached beacons that have been collected from a scan.	
Command Parameters: None	
Response Parameters: None	

Name: setLogicalAndRadioChannel	**ID:** 0x00B9
Description: This call sets the radio channel in the stack and propagates the information to the hardware.	
Command Parameters:	
uint8_t radioChannel	The radio channel to be set.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: setLogicalChannel	**ID:** 0x00BA
Description: Get the logical channel from the ZLL stack.	
Command Parameters: None	
Response Parameters:	
uint8_t logicalChannel	

7 Binding Frames

Name: clearBindingTable	**ID:** 0x002A
Description: Deletes all binding table entries.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: setBinding	**ID:** 0x002B
Description: Sets an entry in the binding table.	
Command Parameters:	
uint8_t index	The index of a binding table entry.
EmberBindingTableEntry value	The contents of the binding entry.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: getBinding	**ID:** 0x002C
Description: Gets an entry from the binding table.	
Command Parameters:	
uint8_t index	The index of a binding table entry.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
EmberBindingTableEntry value	The contents of the binding entry.

Name: deleteBinding	**ID:** 0x002D
Description: Deletes a binding table entry.	
Command Parameters:	
uint8_t index	The index of a binding table entry.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: bindingIsActive	**ID:** 0x002E
Description: Indicates whether any messages are currently being sent using this binding table entry. Note that this command does not indicate whether a binding is clear. To determine whether a binding is clear, check whether the type field of the EmberBindingTableEntry has the value EMBER_UNUSED_BINDING.	
Command Parameters:	
uint8_t index	The index of a binding table entry.
Response Parameters:	
bool active	True if the binding table entry is active, false otherwise.

Name: getBindingRemoteNodeId	**ID:** 0x002F
Description: Returns the node ID for the binding's destination, if the ID is known. If a message is sent using the binding and the destination's ID is not known, the stack will discover the ID by broadcasting a ZDO address request. The application can avoid the need for this discovery by using *setBindingRemoteNodeId* when it knows the correct ID via some other means. The destination's node ID is forgotten when the binding is changed, when the local node reboots or, much more rarely, when the destination node changes its ID in response to an ID conflict.	
Command Parameters:	
uint8_t index	The index of a binding table entry.
Response Parameters:	
EmberNodeId nodeId	The short ID of the destination node or EMBER_NULL_NODE_ID if no destination is known.

Name: setBindingRemoteNodeId	**ID:** 0x0030
Description: Set the node ID for the binding's destination. See *getBindingRemoteNodeId* for a description.	
Command Parameters:	
uint8_t index	The index of a binding table entry.
EmberNodeId nodeId	The short ID of the destination node.
Response Parameters: None	

Name: remoteSetBindingHandler	**ID:** 0x0031
Description: The NCP used the external binding modification policy to decide how to handle a remote set binding request. The Host cannot change the current decision, but it can change the policy for future decisions using the *setPolicy* command.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberBindingTableEntry entry	The requested binding.
uint8_t index	The index at which the binding was added.
EmberStatus policyDecision	EMBER_SUCCESS if the binding was added to the table and any other status if not.

Name: remoteDeleteBindingHandler	**ID:** 0x0032
Description: The NCP used the external binding modification policy to decide how to handle a remote delete binding request. The Host cannot change the current decision, but it can change the policy for future decisions using the *setPolicy* command.	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t index	The index of the binding whose deletion was requested.
EmberStatus policyDecision	EMBER_SUCCESS if the binding was removed from the table and any other status if not.

8 Messaging Frames

Name: maximumPayloadLength	**ID:** 0x0033
Description: Returns the maximum size of the payload. The size depends on the security level in use.	
Command Parameters: None	
Response Parameters:	
uint8_t apsLength	The maximum APS payload length.

Name: sendUnicast	**ID:** 0x0034
Description: Sends a unicast message as per the ZigBee specification. The message will arrive at its destination only if there is a known route to the destination node. Setting the ENABLE_ROUTE_DISCOVERY option will cause a route to be discovered if none is known. Setting the FORCE_ROUTE_DISCOVERY option will force route discovery. Routes to end-device children of the local node are always known. Setting the APS_RETRY option will cause the message to be retransmitted until either a matching acknowledgement is received or three transmissions have been made. **Note:** Using the FORCE_ROUTE_DISCOVERY option will cause the first transmission to be consumed by a route request as part of discovery, so the application payload of this packet will not reach its destination on the first attempt. If you want the packet to reach its destination, the APS_RETRY option must be set so that another attempt is made to transmit the message with its application payload after the route has been constructed. **Note:** When sending fragmented messages, the stack will only assign a new APS sequence number for the first fragment of the message (i.e., EMBER_APS_OPTION_FRAGMENT is set and the low-order byte of the groupId field in the APS frame is zero). For all subsequent fragments of the same message, the application must set the sequence number field in the APS frame to the sequence number assigned by the stack to the first fragment.	
Command Parameters:	
EmberOutgoingMessageType type	Specifies the outgoing message type. Must be one of EMBER_OUTGOING_DIRECT, EMBER_OUTGOING_VIA_ADDRESS_TABLE, or EMBER_OUTGOING_VIA_BINDING.
EmberNodeId indexOrDestination	Depending on the type of addressing used, this is either the EmberNodeId of the destination, an index into the address table, or an index into the binding table.
EmberApsFrame apsFrame	The APS frame which is to be added to the message.
uint8_t messageTag	A value chosen by the Host. This value is used in the *ezspMessageSentHandler* response to refer to this message.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	Content of the message.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
uint8_t sequence	The sequence number that will be used when this message is transmitted.

Name: sendBroadcast	**ID:** 0x0036
Description: Sends a broadcast message as per the ZigBee specification.	
Command Parameters:	
EmberNodeId destination	The destination to which to send the broadcast. This must be one of the three ZigBee broadcast addresses.
EmberApsFrame apsFrame	The APS frame for the message.
uint8_t radius	The message will be delivered to all nodes within *radius* hops of the sender. A radius of zero is converted to EMBER_MAX_HOPS.
uint8_t messageTag	A value chosen by the Host. This value is used in the *ezspMessageSentHandler* response to refer to this message.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The broadcast message.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
uint8_t sequence	The sequence number that will be used when this message is transmitted.

Name: proxyBroadcast	**ID:** 0x0037
Description: Sends a proxied broadcast message as per the ZigBee specification.	
Command Parameters:	
EmberNodeId source	The source from which to send the broadcast.
EmberNodeId destination	The destination to which to send the broadcast. This must be one of the three ZigBee broadcast addresses.
uint8_t nwkSequence	The network sequence number for the broadcast.
EmberApsFrame apsFrame	The APS frame for the message.
uint8_t radius	The message will be delivered to all nodes within *radius* hops of the sender. A radius of zero is converted to EMBER_MAX_HOPS.
uint8_t messageTag	A value chosen by the Host. This value is used in the *ezspMessageSentHandler* response to refer to this message.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The broadcast message.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
uint8_t apsSequence	The APS sequence number that will be used when this message is transmitted.

Name: sendMulticast	**ID:** 0x0038
Description: Sends a multicast message to all endpoints that share a specific multicast ID and are within a specified number of hops of the sender.	
Command Parameters:	
EmberApsFrame apsFrame	The APS frame for the message. The multicast will be sent to the groupId in this frame.
uint8_t hops	The message will be delivered to all nodes within this number of hops of the sender. A value of zero is converted to EMBER_MAX_HOPS.
uint8_t nonmemberRadius	The number of hops that the message will be forwarded by devices that are not members of the group. A value of 7 or greater is treated as infinite.
uint8_t messageTag	A value chosen by the Host. This value is used in the *ezspMessageSentHandler* response to refer to this message.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The multicast message.
Response Parameters:	
EmberStatus status	An EmberStatus value. For any result other than EMBER_SUCCESS, the message will not be sent. EMBER_SUCCESS - The message has been submitted for transmission. EMBER_INVALID_BINDING_INDEX - The bindingTableIndex refers to a non-multicast binding. EMBER_NETWORK_DOWN - The node is not part of a network. EMBER_MESSAGE_TOO_LONG - The message is too large to fit in a MAC layer frame. EMBER_NO_BUFFERS - The free packet buffer pool is empty. EMBER_NETWORK_BUSY - Insufficient resources available in Network or MAC layers to send message.
uint8_t sequence	The sequence number that will be used when this message is transmitted.

Name: sendMulticastWithAlias	**ID:** 0x003A
Description: Sends a multicast message to all endpoints that share a specific multicast ID and are within a specified number of hops of the sender.	
Command Parameters:	
EmberApsFrame apsFrame	The APS frame for the message. The multicast will be sent to the groupId in this frame.
uint8_t hops	The message will be delivered to all nodes within this number of hops of the sender. A value of zero is converted to EMBER_MAX_HOPS.
uint8_t nonmemberRadius	The number of hops that the message will be forwarded by devices that are not members of the group. A value of 7 or greater is treated as infinite.
uint16_t alias	The alias source address
uint8_t nwkSequence	the alias sequence number
uint8_t messageTag	A value chosen by the Host. This value is used in the *ezspMessageSentHandler* response to refer to this message.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The multicast message.
Response Parameters:	
EmberStatus status	An EmberStatus value. For any result other than EMBER_SUCCESS, the message will not be sent. EMBER_SUCCESS - The message has been submitted for transmission. EMBER_INVALID_BINDING_INDEX - The bindingTableIndex refers to a non-multicast binding. EMBER_NETWORK_DOWN - The node is not part of a network. EMBER_MESSAGE_TOO_LONG - The message is too large to fit in a MAC layer frame. EMBER_NO_BUFFERS - The free packet buffer pool is empty. EMBER_NETWORK_BUSY - Insufficient resources available in Network or MAC layers to send message.
uint8_t sequence	The sequence number that will be used when this message is transmitted.

Name: sendReply	**ID:** 0x0039
Description: Sends a reply to a received unicast message. The *incomingMessageHandler* callback for the unicast being replied to supplies the values for all the parameters except the reply itself.	
Command Parameters:	
EmberNodeId sender	Value supplied by incoming unicast.
EmberApsFrame apsFrame	Value supplied by incoming unicast.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The reply message.
Response Parameters:	
EmberStatus status	An EmberStatus value. EMBER_INVALID_CALL - The EZSP_UNICAST_REPLIES_POLICY is set to EZSP_HOST_WILL_NOT_SUPPLY_REPLY. This means the NCP will automatically send an empty reply. The Host must change the policy to EZSP_HOST_WILL_SUPPLY_REPLY before it can supply the reply. There is one exception to this rule: In the case of responses to message fragments, the host must call sendReply when a message fragment is received. In this case, the policy set on the NCP does not matter. The NCP expects a sendReply call from the Host for message fragments regardless of the current policy settings. EMBER_NO_BUFFERS - Not enough memory was available to send the reply. EMBER_NETWORK_BUSY - Either no route or insufficient resources available. EMBER_SUCCESS - The reply was successfully queued for transmission.

Name: messageSentHandler	**ID:** 0x003F
Description: A callback indicating the stack has completed sending a message.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberOutgoingMessageType type	The type of message sent.
uint16_t indexOrDestination	The destination to which the message was sent, for direct unicasts, or the address table or binding index for other unicasts. The value is unspecified for multicasts and broadcasts.
EmberApsFrame apsFrame	The APS frame for the message.
uint8_t messageTag	The value supplied by the Host in the *ezspSendUnicast*, *ezspSendBroadcast* or *ezspSendMulticast* command.
EmberStatus status	An EmberStatus value of EMBER_SUCCESS if an ACK was received from the destination or EMBER_DELIVERY_FAILED if no ACK was received.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The unicast message supplied by the Host. The message contents are only included here if the decision for the messageContentsInCallback policy is messageTagAndContentsInCallback.

Name: sendManyToOneRouteRequest **ID:** 0x0041

Description: Sends a route request packet that creates routes from every node in the network back to this node. This function should be called by an application that wishes to communicate with many nodes, for example, a gateway, central monitor, or controller. A device using this function was referred to as an 'aggregator' in EmberZNet 2.x and earlier, and is referred to as a 'concentrator' in the ZigBee specification and EmberZNet 3.

This function enables large scale networks, because the other devices do not have to individually perform bandwidth-intensive route discoveries. Instead, when a remote node sends an APS unicast to a concentrator, its network layer automatically delivers a special route record packet first, which lists the network ids of all the intermediate relays. The concentrator can then use source routing to send outbound APS unicasts. (A source routed message is one in which the entire route is listed in the network layer header.) This allows the concentrator to communicate with thousands of devices without requiring large route tables on neighboring nodes.

This function is only available in ZigBee Pro (stack profile 2), and cannot be called on end devices. Any router can be a concentrator (not just the coordinator), and there can be multiple concentrators on a network.

Note that a concentrator does not automatically obtain routes to all network nodes after calling this function. Remote applications must first initiate an inbound APS unicast.

Many-to-one routes are not repaired automatically. Instead, the concentrator application must call this function to rediscover the routes as necessary, for example, upon failure of a retried APS message. The reason for this is that there is no scalable one-size-fits-all route repair strategy. A common and recommended strategy is for the concentrator application to refresh the routes by calling this function periodically.

Command Parameters:

uint16_t concentratorType	Must be either EMBER_HIGH_RAM_CONCENTRATOR or EMBER_LOW_RAM_CONCENTRATOR. The former is used when the caller has enough memory to store source routes for the whole network. In that case, remote nodes stop sending route records once the concentrator has successfully received one. The latter is used when the concentrator has insufficient RAM to store all outbound source routes. In that case, route records are sent to the concentrator prior to every inbound APS unicast.
uint8_t radius	The maximum number of hops the route request will be relayed. A radius of zero is converted to EMBER_MAX_HOPS.

Response Parameters:

EmberStatus status	EMBER_SUCCESS if the route request was successfully submitted to the transmit queue, and EMBER_ERR_FATAL otherwise.

Name: pollForData	**ID:** 0x0042
Description: Periodically request any pending data from our parent. Setting *interval* to 0 or *units* to EMBER_EVENT_INACTIVE will generate a single poll.	
Command Parameters:	
uint16_t interval	The time between polls. Note that the timer clock is free running and is not synchronized with this command. This means that the time will be between *interval* and (*interval* - 1). The maximum interval is 32767.
EmberEventUnits units	The units for *interval*.
uint8_t failureLimit	The number of poll failures that will be tolerated before a *pollCompleteHandler* callback is generated. A value of zero will result in a callback for every poll. Any status value apart from EMBER_SUCCESS and EMBER_MAC_NO_DATA is counted as a failure.
Response Parameters:	
EmberStatus status	The result of sending the first poll.

Name: pollCompleteHandler	**ID:** 0x0043
Description: Indicates the result of a data poll to the parent of the local node.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	An EmberStatus value: EMBER_SUCCESS - Data was received in response to the poll. EMBER_MAC_NO_DATA - No data was pending. EMBER_DELIVERY_FAILED - The poll message could not be sent. EMBER_MAC_NO_ACK_RECEIVED - The poll message was sent but not acknowledged by the parent.

Name: pollHandler	**ID:** 0x0044
Description: Indicates that the local node received a data poll from a child.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberNodeId childId	The node ID of the child that is requesting data.
bool transmitExpected	True if transmit is expected, false otherwise.

Name: incomingSenderEui64Handler	**ID:** 0x0062
Description: A callback indicating a message has been received containing the EUI64 of the sender. This callback is called immediately before the *incomingMessageHandler* callback. It is not called if the incoming message did not contain the EUI64 of the sender.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberEUI64 senderEui64	The EUI64 of the sender

Name: incomingMessageHandler	**ID:** 0x0045
Description: A callback indicating a message has been received.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberIncomingMessageType type	The type of the incoming message. One of the following: EMBER_INCOMING_UNICAST, EMBER_INCOMING_UNICAST_REPLY, EMBER_INCOMING_MULTICAST, EMBER_INCOMING_MULTICAST_LOOPBACK, EMBER_INCOMING_BROADCAST, EMBER_INCOMING_BROADCAST_LOOPBACK
EmberApsFrame apsFrame	The APS frame from the incoming message.
uint8_t lastHopLqi	The link quality from the node that last relayed the message.
int8s lastHopRssi	The energy level (in units of dBm) observed during the reception.
EmberNodeId sender	The sender of the message.
uint8_t bindingIndex	The index of a binding that matches the message or 0xFF if there is no matching binding.
uint8_t addressIndex	The index of the entry in the address table that matches the sender of the message or 0xFF if there is no matching entry.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The incoming message.

Name: setSourceRouteDiscoveryMode	**ID:** 0x005A
Description: Sets source route discovery(MTORR) mode to on, off, reschedule	
Command Parameters:	
uint8_t mode	Source route discovery mode: off:0, on:1, reschedule:2
Response Parameters:	
uint32_t remainingTime	Remaining time(ms) until next MTORR broadcast if the mode is on, MAX_INT32U_VALUE if the mode is off

Name: incomingManyToOneRouteRequestHandler	**ID:** 0x007D
Description: A callback indicating that a many-to-one route to the concentrator with the given short and long id is available for use.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberNodeId source	The short id of the concentrator.
EmberEUI64 longId	The EUI64 of the concentrator.
uint8_t cost	The path cost to the concentrator. The cost may decrease as additional route request packets for this discovery arrive, but the callback is made only once.

Name: incomingRouteErrorHandler	**ID:** 0x0080
Description: A callback invoked when a route error message is received. The error indicates that a problem routing to or from the target node was encountered.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	EMBER_SOURCE_ROUTE_FAILURE or EMBER_MANY_TO_ONE_ROUTE_FAILURE.
EmberNodeId target	The short id of the remote node.

Name: incomingNetworkStatusHandler	**ID:** 0x00C4
Description: A callback invoked when a network status/route error message is received. The error indicates that there was a problem sending/receiving messages from the target node	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t errorCode	One byte over-the-air error code from network status message
EmberNodeId target	The short ID of the remote node

Name: unicastCurrentNetworkKey	**ID:** 0x0050
Description: Send the network key to a destination.	
Command Parameters:	
EmberNodeId targetShort	The destination node of the key.
EmberEUI64 targetLong	The long address of the destination node.
EmberNodeId parentShortId	The parent node of the destination node.
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if send was successful

Name: addressTableEntryIsActive	**ID:** 0x005B
Description: Indicates whether any messages are currently being sent using this address table entry. Note that this function does not indicate whether the address table entry is unused. To determine whether an address table entry is unused, check the remote node ID. The remote node ID will have the value EMBER_TABLE_ENTRY_UNUSED_NODE_ID when the address table entry is not in use.	
Command Parameters:	
uint8_t addressTableIndex	The index of an address table entry.
Response Parameters:	
bool active	True if the address table entry is active, false otherwise.

Name: setAddressTableRemoteEui64	**ID:** 0x005C
Description: Sets the EUI64 of an address table entry. This function will also check other address table entries, the child table and the neighbor table to see if the node ID for the given EUI64 is already known. If known then this function will also set node ID. If not known it will set the node ID to EMBER_UNKNOWN_NODE_ID.	
Command Parameters:	
uint8_t addressTableIndex	The index of an address table entry.
EmberEUI64 eui64	The EUI64 to use for the address table entry.
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if the EUI64 was successfully set, and EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE otherwise.

Name: setAddressTableRemoteNodeId	**ID:** 0x005D
Description: Sets the short ID of an address table entry. Usually the application will not need to set the short ID in the address table. Once the remote EUI64 is set the stack is capable of figuring out the short ID on its own. However, in cases where the application does set the short ID, the application must set the remote EUI64 prior to setting the short ID.	
Command Parameters:	
uint8_t addressTableIndex	The index of an address table entry.
EmberNodeId id	The short ID corresponding to the remote node whose EUI64 is stored in the address table at the given index or EMBER_TABLE_ENTRY_UNUSED_NODE_ID which indicates that the entry stored in the address table at the given index is not in use.
Response Parameters: None	

Name: getAddressTableRemoteEui64	**ID:** 0x005E
Description: Gets the EUI64 of an address table entry.	
Command Parameters:	
uint8_t addressTableIndex	The index of an address table entry.
Response Parameters:	
EmberEUI64 eui64	The EUI64 of the address table entry is copied to this location.

Name: getAddressTableRemoteNodeId	**ID:** 0x005F
Description: Gets the short ID of an address table entry.	
Command Parameters:	
uint8_t addressTableIndex	The index of an address table entry.
Response Parameters:	
EmberNodeId nodeId	One of the following: The short ID corresponding to the remote node whose EUI64 is stored in the address table at the given index. EMBER_UNKNOWN_NODE_ID - Indicates that the EUI64 stored in the address table at the given index is valid but the short ID is currently unknown. EMBER_DISCOVERY_ACTIVE_NODE_ID - Indicates that the EUI64 stored in the address table at the given location is valid and network address discovery is underway. EMBER_TABLE_ENTRY_UNUSED_NODE_ID - Indicates that the entry stored in the address table at the given index is not in use.

Name: setExtendedTimeout	**ID:** 0x007E
Description: Tells the stack whether or not the normal interval between retransmissions of a retried unicast message should be increased by EMBER_INDIRECT_TRANSMISSION_TIMEOUT. The interval needs to be increased when sending to a sleepy node so that the message is not retransmitted until the destination has had time to wake up and poll its parent. The stack will automatically extend the timeout: - For our own sleepy children. - When an address response is received from a parent on behalf of its child. - When an indirect transaction expiry route error is received. - When an end device announcement is received from a sleepy node.	
Command Parameters:	
EmberEUI64 remoteEui64	The address of the node for which the timeout is to be set.
bool extendedTimeout	true if the retry interval should be increased by EMBER_INDIRECT_TRANSMISSION_TIMEOUT. false if the normal retry interval should be used.
Response Parameters: None	

Name: getExtendedTimeout	**ID:** 0x007F
Description: Indicates whether or not the stack will extend the normal interval between retransmissions of a retried unicast message by EMBER_INDIRECT_TRANSMISSION_TIMEOUT.	
Command Parameters:	
EmberEUI64 remoteEui64	The address of the node for which the timeout is to be returned.
Response Parameters:	
bool extendedTimeout	true if the retry interval will be increased by EMBER_INDIRECT_TRANSMISSION_TIMEOUT and false if the normal retry interval will be used.

Name: replaceAddressTableEntry	**ID:** 0x0082
Description: Replaces the EUI64, short ID and extended timeout setting of an address table entry. The previous EUI64, short ID and extended timeout setting are returned.	
Command Parameters:	
uint8_t addressTableIndex	The index of the address table entry that will be modified.
EmberEUI64 newEui64	The EUI64 to be written to the address table entry.
EmberNodeId newId	One of the following: The short ID corresponding to the new EUI64. EMBER_UNKNOWN_NODE_ID if the new EUI64 is valid but the short ID is unknown and should be discovered by the stack. EMBER_TABLE_ENTRY_UNUSED_NODE_ID if the address table entry is now unused.
bool newExtendedTimeout	true if the retry interval should be increased by EMBER_INDIRECT_TRANSMISSION_TIMEOUT. false if the normal retry interval should be used.
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if the EUI64, short ID and extended timeout setting were successfully modified, and EMBER_ADDRESS_TABLE_ENTRY_IS_ACTIVE otherwise.
EmberEUI64 oldEui64	The EUI64 of the address table entry before it was modified.
EmberNodeId oldId	One of the following: The short ID corresponding to the EUI64 before it was modified. EMBER_UNKNOWN_NODE_ID if the short ID was unknown. EMBER_DISCOVERY_ACTIVE_NODE_ID if discovery of the short ID was underway. EMBER_TABLE_ENTRY_UNUSED_NODE_ID if the address table entry was unused.
bool oldExtendedTimeout	true if the retry interval was being increased by EMBER_INDIRECT_TRANSMISSION_TIMEOUT. false if the normal retry interval was being used.

Name: lookupNodeIdByEui64	**ID:** 0x0060
Description: Returns the node ID that corresponds to the specified EUI64. The node ID is found by searching through all stack tables for the specified EUI64.	
Command Parameters:	
EmberEUI64 eui64	The EUI64 of the node to look up.
Response Parameters:	
EmberNodeId nodeId	The short ID of the node or EMBER_NULL_NODE_ID if the short ID is not known.

Name: lookupEui64ByNodeId	**ID:** 0x0061
Description: Returns the EUI64 that corresponds to the specified node ID. The EUI64 is found by searching through all stack tables for the specified node ID.	
Command Parameters:	
EmberNodeId nodeId	The short ID of the node to look up.
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if the EUI64 was found, EMBER_ERR_FATAL if the EUI64 is not known.
EmberEUI64 eui64	The EUI64 of the node.

Name: getMulticastTableEntry	**ID:** 0x0063
Description: Gets an entry from the multicast table.	
Command Parameters:	
uint8_t index	The index of a multicast table entry.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
EmberMulticastTableEntry value	The contents of the multicast entry.

Name: setMulticastTableEntry	**ID:** 0x0064
Description: Sets an entry in the multicast table.	
Command Parameters:	
uint8_t index	The index of a multicast table entry
EmberMulticastTableEntry value	The contents of the multicast entry.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: idConflictHandler	**ID:** 0x007C
Description: A callback invoked by the EmberZNet stack when an id conflict is discovered, that is, two different nodes in the network were found to be using the same short id. The stack automatically removes the conflicting short id from its internal tables (address, binding, route, neighbor, and child tables). The application should discontinue any other use of the id.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberNodeId id	The short id for which a conflict was detected

Name: writeNodeData	**ID:** 0x00FE
Description: Write the current node Id, PAN ID, or Node type to the tokens	
Command Parameters:	
bool erase	Erase the node type or not
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: sendRawMessage	**ID:** 0x0096
Description: Transmits the given message without modification. The MAC header is assumed to be configured in the message at the time this function is called.	
Command Parameters:	
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The raw message.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: sendRawMessageExtended	**ID:** 0x0051
Description: Transmits the given message without modification. The MAC header is assumed to be configured in the message at the time this function is called.	
Command Parameters:	
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The raw message.
uint8_t priority	transmit priority.
bool useCca	Should we enable CCA or not.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: macPassthroughMessageHandler	**ID:** 0x0097
Description: A callback invoked by the EmberZNet stack when a MAC passthrough message is received.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberMacPassthroughType messageType	The type of MAC passthrough message received.
uint8_t lastHopLqi	The link quality from the node that last relayed the message.
int8s lastHopRssi	The energy level (in units of dBm) observed during reception.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The raw message that was received.

Name: macFilterMatchMessageHandler	**ID:** 0x0046
Description: A callback invoked by the EmberZNet stack when a raw MAC message that has matched one of the application's configured MAC filters.	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t filterIndexMatch	The index of the filter that was matched.
EmberMacPassthroughType legacyPassthroughType	The type of MAC passthrough message received.
uint8_t lastHopLqi	The link quality from the node that last relayed the message.
int8s lastHopRssi	The energy level (in units of dBm) observed during reception.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The raw message that was received.

Name: rawTransmitCompleteHandler	**ID:** 0x0098
Description: A callback invoked by the EmberZNet stack when the MAC has finished transmitting a raw message.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	EMBER_SUCCESS if the transmission was successful, or EMBER_DELIVERY_FAILED if not

Name: setMacPollFailureWaitTime	**ID:** 0x00F4
Description: This function is useful to sleepy end devices. This function will set the retry interval (in milliseconds) for mac data poll. This interval is the time in milliseconds the device waits before retrying a data poll when a MAC level data poll fails for any reason.	
Command Parameters:	
uint8_t waitBeforeRetryIntervalMs	Time in seconds the device waits before retrying a data poll when a MAC level data poll fails for any reason.
Response Parameters: None	

Name: setBeaconClassificationParams	**ID:** 0x00EF
Description: Sets the priority masks and related variables for choosing the best beacon.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	The attempt to set the parameters returns EMBER_SUCCESS.
EmberBeaconClassificationParams param	Gets the beacon prioritization related variable.

Name: getBeaconClassificationParams	**ID:** 0x00F3
Description: Gets the priority masks and related variables for choosing the best beacon.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	The attempt to get the parameters returns EMBER_SUCCESS.
EmberBeaconClassificationParams param	Gets the beacon prioritization related variable.

9 Security Frames

Name: setInitialSecurityState	**ID:** 0x0068
Description: Sets the security state that will be used by the device when it forms or joins the network. This call **should not** be used when restoring saved network state via networkInit as this will result in a loss of security data and will cause communication problems when the device re-enters the network.	
Command Parameters:	
EmberInitialSecurityState state	The security configuration to be set.
Response Parameters:	
EmberStatus success	The success or failure code of the operation.

Name: getCurrentSecurityState	**ID:** 0x0069
Description: Gets the current security state that is being used by a device that is joined in the network.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	The success or failure code of the operation.
EmberCurrentSecurityState state	The security configuration in use by the stack.

Name: exportKey	**ID:** 0x0114
Description: Exports a key from security manager based on passed context.	
Command Parameters:	
sl_zb_sec_man_context_t context	Metadata to identify the requested key.
Response Parameters:	
sl_zb_sec_man_key_t key	Data to store the exported key in.
sl_status_t status	The success or failure code of the operation.

Name: importKey	**ID:** 0x0115
Description: Imports a key into security manager based on passed context.	
Command Parameters:	
sl_zb_sec_man_context_t context	Metadata to identify where the imported key should be stored.
sl_zb_sec_man_key_t key	The key to be imported.
Response Parameters:	
sl_status_t status	The success or failure code of the operation.

Name: switchNetworkKeyHandler	**ID:** 0x006e
Description: A callback to inform the application that the Network Key has been updated and the node has been switched over to use the new key. The actual key being used is not passed up, but the sequence number is.	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t sequenceNumber	The sequence number of the new network key.

Name: findKeyTableEntry	**ID:** 0x0075
Description: This function searches through the Key Table and tries to find the entry that matches the passed search criteria.	
Command Parameters:	
EmberEUI64 address	The address to search for. Alternatively, all zeros may be passed in to search for the first empty entry.
bool linkKey	This indicates whether to search for an entry that contains a link key or a master key. true means to search for an entry with a Link Key.
Response Parameters:	
uint8_t index	This indicates the index of the entry that matches the search criteria. A value of 0x00FF is returned if not matching entry is found.

Name: sendTrustCenterLinkKey	**ID:** 0x0067
Description: This function sends an APS TransportKey command containing the current trust center link key. The node to which the command is sent is specified via the short and long address arguments.	
Command Parameters:	
EmberNodeId destinationNodeId	The short address of the node to which this command will be sent
EmberEUI64 destinationEui64	The long address of the node to which this command will be sent
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success of failure of the operation

Name: eraseKeyTableEntry	**ID:** 0x0076
Description: This function erases the data in the key table entry at the specified index. If the index is invalid, false is returned.	
Command Parameters:	
uint8_t index	This indicates the index of entry to erase.
Response Parameters:	
EmberStatus status	The success or failure of the operation.

Name: clearKeyTable	**ID:** 0x00B1
Description: This function clears the key table of the current network.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	The success or failure of the operation.

Name: requestLinkKey	**ID:** 0x0014
Description: A function to request a Link Key from the Trust Center with another device on the Network (which could be the Trust Center). A Link Key with the Trust Center is possible but the requesting device cannot be the Trust Center. Link Keys are optional in ZigBee Standard Security and thus the stack cannot know whether the other device supports them. If EMBER_REQUEST_KEY_TIMEOUT is non-zero on the Trust Center and the partner device is not the Trust Center, both devices must request keys with their partner device within the time period. The Trust Center only supports one outstanding key request at a time and therefore will ignore other requests. If the timeout is zero then the Trust Center will immediately respond and not wait for the second request. The Trust Center will always immediately respond to requests for a Link Key with it. Sleepy devices should poll at a higher rate until a response is received or the request times out. The success or failure of the request is returned via ezspZigbeeKeyEstablishmentHandler(...).	
Command Parameters:	
EmberEUI64 partner	This is the IEEE address of the partner device that will share the link key.
Response Parameters:	
EmberStatus status	The success or failure of sending the request. This is not the final result of the attempt. ezspZigbeeKeyEstablishmentHandler(...) will return that.

Name: updateTcLinkKey	**ID:** 0x006C
Description: Requests a new link key from the Trust Center. This function starts by sending a Node Descriptor request to the Trust Center to verify its R21+ stack version compliance. A Request Key message will then be sent, followed by a Verify Key Confirm message.	
Command Parameters:	
uint8_t maxAttempts	The maximum number of attempts a node should make when sending the Node Descriptor, Request Key, and Verify Key Confirm messages. The number of attempts resets for each message type sent (e.g., if maxAttempts is 3, up to 3 Node Descriptors are sent, up to 3 Request Keys, and up to 3 Verify Key Confirm messages are sent).
Response Parameters:	
EmberStatus status	The success or failure of sending the request. If the Node Descriptor is successfully transmitted, ezspZigbeeKeyEstablishmentHandler(...) will be called at a later time with a final status result.

Name: zigbeeKeyEstablishmentHandler	**ID:** 0x009B
Description: This is a callback that indicates the success or failure of an attempt to establish a key with a partner device.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberEUI64 partner	This is the IEEE address of the partner that the device successfully established a key with. This value is all zeros on a failure.
EmberKeyStatus status	This is the status indicating what was established or why the key establishment failed.

Name: clearTransientLinkKeys	**ID:** 0x006B
Description: Clear all of the transient link keys from RAM.	
Command Parameters: None	
Response Parameters: None	

Name: getNetworkKeyInfo	**ID:** 0x0116
Description: Retrieve information about the current and alternate network key, excluding their contents.	
Command Parameters: None	
Response Parameters:	
sl_status_t status	Success or failure of retrieving network key info.
sl_zb_sec_man_network_key_info_t network_key_info	Information about current and alternate network keys.

Name: getApsKeyInfo	**ID:** 0x010C
Description: Retrieve metadata about an APS link key. Does not retrieve contents.	
Command Parameters:	
sl_zb_sec_man_context_t context_in	Context used to input information about key.
Response Parameters:	
EmberEUI64 eui	EUI64 associated with this APS link key
sl_zb_sec_man_aps_key_metadata_t key_data	Metadata about the referenced key.
sl_status_t status	Status of metadata retrieval operation.

Name: importLinkKey	**ID:** 0x010E
Description: Import an application link key into the key table.	
Command Parameters:	
uint8_t index	Index where this key is to be imported to.
EmberEUI64 address	EUI64 this key is associated with.
sl_zb_sec_man_key_t plaintext_key	The key data to be imported.
Response Parameters:	
sl_status_t status	Status of key import operation.

Name: exportLinkKeyByIndex	**ID:** 0x010F
Description: Export the link key at given index from the key table.	
Command Parameters:	
uint8_t index	Index of key to export.
Response Parameters:	
EmberEUI64 eui	EUI64 associated with the exported key.
sl_zb_sec_man_key_t plaintext_key	The exported key.
sl_zb_sec_man_aps_key_metadata_t key_data	Metadata about the key.
sl_status_t status	Status of key export operation.

Name: exportLinkKeyByEui	**ID:** 0x010D
Description: Export the link key associated with the given EUI from the key table.	
Command Parameters:	
EmberEUI64 eui	EUI64 associated with the key to export.
Response Parameters:	
sl_zb_sec_man_key_t plaintext_key	The exported key.
uint8_t index	Key index of the exported key.
sl_zb_sec_man_aps_key_metadata_t key_data	Metadata about the key.
sl_status_t status	Status of key export operation.

Name: checkKeyContext	**ID:** 0x0110
Description: Check whether a key context can be used to load a valid key.	
Command Parameters:	
sl_zb_sec_man_context_t context	Context struct to check the validity of.
Response Parameters:	
sl_status_t status	Validity of the checked context.

Name: importTransientKey	**ID:** 0x0111
Description: Import a transient link key.	
Command Parameters:	
EmberEUI64 eui64	EUI64 associated with this transient key.
sl_zb_sec_man_key_t plaintext_key	The key to import.
sl_zigbee_sec_man_flags_t flags	Flags associated with this transient key.
Response Parameters:	
sl_status_t status	Status of key import operation.

Name: exportTransientKeyByIndex	**ID:** 0x0112
Description: Export a transient link key from a given table index.	
Command Parameters:	
uint8_t index	Index to export from.
Response Parameters:	
sl_zb_sec_man_context_t context	Context struct for export operation.
sl_zb_sec_man_key_t plaintext_key	The exported key.
sl_zb_sec_man_aps_key_metadata_t key_data	Metadata about the key.
sl_status_t status	Status of key export operation.

Name: exportTransientKeyByEui	**ID:** 0x0113
Description: Export a transient link key associated with a given EUI64	
Command Parameters:	
EmberEUI64 eui	Index to export from.
Response Parameters:	
sl_zb_sec_man_context_t context	Context struct for export operation.
sl_zb_sec_man_key_t plaintext_key	The exported key.
sl_zb_sec_man_aps_key_metadata_t key_data	Metadata about the key.
sl_status_t status	Status of key export operation.

10 Trust Center Frames

Name: trustCenterJoinHandler	**ID:** 0x0024
Description: The NCP used the trust center behavior policy to decide whether to allow a new node to join the network. The Host cannot change the current decision, but it can change the policy for future decisions using the *setPolicy* command.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberNodeId newNodeId	The Node Id of the node whose status changed
EmberEUI64 newNodeEui64	The EUI64 of the node whose status changed.
EmberDeviceUpdate status	The status of the node: Secure Join/Rejoin, Unsecure Join/Rejoin, Device left.
EmberJoinDecision policyDecision	An EmberJoinDecision reflecting the decision made.
EmberNodeId parentOfNewNodeId	The parent of the node whose status has changed.

Name: broadcastNextNetworkKey	**ID:** 0x0073
Description: This function broadcasts a new encryption key, but does not tell the nodes in the network to start using it. To tell nodes to switch to the new key, use emberSendNetworkKeySwitch(). This is only valid for the Trust Center/Coordinator. It is up to the application to determine how quickly to send the Switch Key after sending the alternate encryption key.	
Command Parameters:	
EmberKeyData key	An optional pointer to a 16-byte encryption key (EMBER_ENCRYPTION_KEY_SIZE). An all zero key may be passed in, which will cause the stack to randomly generate a new key.
Response Parameters:	
EmberStatus status	EmberStatus value that indicates the success or failure of the command.

Name: broadcastNetworkKeySwitch	**ID:** 0x0074
Description: This function broadcasts a switch key message to tell all nodes to change to the sequence number of the previously sent Alternate Encryption Key.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	EmberStatus value that indicates the success or failure of the command.

Name: aesMmoHash	**ID:** 0x006F
Description: This routine processes the passed chunk of data and updates the hash context based on it. If the 'finalize' parameter is not set, then the length of the data passed in must be a multiple of 16. If the 'finalize' parameter is set then the length can be any value up 1-16, and the final hash value will be calculated.	
Command Parameters:	
EmberAesMmoHashContext context	The hash context to update.
bool finalize	This indicates whether the final hash value should be calculated
uint8_t length	The length of the data to hash.
uint8_t[] data	The data to hash.
Response Parameters:	
EmberStatus status	The result of the operation
EmberAesMmoHashContext returnContext	The updated hash context.

Name: removeDevice	**ID:** 0x00A8
Description: This command sends an APS remove device using APS encryption to the destination indicating either to remove itself from the network, or one of its children.	
Command Parameters:	
EmberNodeId destShort	The node ID of the device that will receive the message
EmberEUI64 destLong	The long address (EUI64) of the device that will receive the message.
EmberEUI64 targetLong	The long address (EUI64) of the device to be removed.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success, or the reason for failure

Name: unicastNwkKeyUpdate	**ID:** 0x00A9
Description: This command will send a unicast transport key message with a new NWK key to the specified device. APS encryption using the device's existing link key will be used.	
Command Parameters:	
EmberNodeId destShort	The node ID of the device that will receive the message
EmberEUI64 destLong	The long address (EUI64) of the device that will receive the message.
EmberKeyData key	The NWK key to send to the new device.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success, or the reason for failure

11 Certificate-Based Key Exchange (CBKE) Frames

Name: generateCbkeKeys	**ID:** 0x00A4
Description: This call starts the generation of the ECC Ephemeral Public/Private key pair. When complete it stores the private key. The results are returned via ezspGenerateCbkeKeysHandler().	
Command Parameters: None	
Response Parameters:	
EmberStatus status	

Name: generateCbkeKeysHandler	**ID:** 0x009E
Description: A callback by the Crypto Engine indicating that a new ephemeral public/private key pair has been generated. The public/private key pair is stored on the NCP, but only the associated public key is returned to the host. The node's associated certificate is also returned.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	The result of the CBKE operation.
EmberPublicKeyData ephemeralPublicKey	The generated ephemeral public key.

Name: calculateSmacs	**ID:** 0x009F
Description: Calculates the SMAC verification keys for both the initiator and responder roles of CBKE using the passed parameters and the stored public/private key pair previously generated with ezspGenerateKeysRetrieveCert(). It also stores the unverified link key data in temporary storage on the NCP until the key establishment is complete.	
Command Parameters:	
bool amInitiator	The role of this device in the Key Establishment protocol.
EmberCertificateData partnerCertificate	The key establishment partner's implicit certificate.
EmberPublicKeyData partnerEphemeralPublicKey	The key establishment partner's ephemeral public key
Response Parameters:	
EmberStatus status	

Name: calculateSmacsHandler	**ID:** 0x00A0
Description: A callback to indicate that the NCP has finished calculating the Secure Message Authentication Codes (SMAC) for both the initiator and responder. The associated link key is kept in temporary storage until the host tells the NCP to store or discard the key via emberClearTemporaryDataMaybeStoreLinkKey().	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	The Result of the CBKE operation.
EmberSmacData initiatorSmac	The calculated value of the initiator's SMAC
EmberSmacData responderSmac	The calculated value of the responder's SMAC

Name: generateCbkeKeys283k1	**ID:** 0x00E8
Description: This call starts the generation of the ECC 283k1 curve Ephemeral Public/Private key pair. When complete it stores the private key. The results are returned via ezspGenerateCbkeKeysHandler283k1().	
Command Parameters: None	
Response Parameters:	
EmberStatus status	

Name: generateCbkeKeysHandler283k1	**ID:** 0x00E9
Description: A callback by the Crypto Engine indicating that a new 283k1 ephemeral public/private key pair has been generated. The public/private key pair is stored on the NCP, but only the associated public key is returned to the host. The node's associated certificate is also returned.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	The result of the CBKE operation.
EmberPublicKey283k1Data ephemeralPublicKey	The generated ephemeral public key.

Name: calculateSmacs283k1	**ID:** 0x00EA
Description: Calculates the SMAC verification keys for both the initiator and responder roles of CBKE for the 283k1 ECC curve using the passed parameters and the stored public/private key pair previously generated with ezspGenerateKeysRetrieveCert283k1(). It also stores the unverified link key data in temporary storage on the NCP until the key establishment is complete.	
Command Parameters:	
bool amInitiator	The role of this device in the Key Establishment protocol.
EmberCertificate283k1Data partnerCertificate	The key establishment partner's implicit certificate.
EmberPublicKey283k1Data partnerEphemeralPublicKey	The key establishment partner's ephemeral public key
Response Parameters:	
EmberStatus status	

Name: calculateSmacsHandler283k1	**ID:** 0x00EB
Description: A callback to indicate that the NCP has finished calculating the Secure Message Authentication Codes (SMAC) for both the initiator and responder for the CBKE 283k1 Library. The associated link key is kept in temporary storage until the host tells the NCP to store or discard the key via emberClearTemporaryDataMaybeStoreLinkKey().	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	The Result of the CBKE operation.
EmberSmacData initiatorSmac	The calculated value of the initiator's SMAC
EmberSmacData responderSmac	The calculated value of the responder's SMAC

Name: clearTemporaryDataMaybeStoreLinkKey	**ID:** 0x00A1
Description: Clears the temporary data associated with CBKE and the key establishment, most notably the ephemeral public/private key pair. If storeLinKey is true it moves the unverified link key stored in temporary storage into the link key table. Otherwise it discards the key.	
Command Parameters:	
bool storeLinkKey	A bool indicating whether to store (true) or discard (false) the unverified link key derived when ezspCalculateSmacs() was previously called.
Response Parameters:	
EmberStatus status	

Name: clearTemporaryDataMaybeStoreLinkKey283k1	**ID:** 0x00EE
Description: Clears the temporary data associated with CBKE and the key establishment, most notably the ephemeral public/private key pair. If storeLinKey is true it moves the unverified link key stored in temporary storage into the link key table. Otherwise it discards the key.	
Command Parameters:	
bool storeLinkKey	A bool indicating whether to store (true) or discard (false) the unverified link key derived when ezspCalculateSmacs() was previously called.
Response Parameters:	
EmberStatus status	

Name: getCertificate	**ID:** 0x00A5
Description: Retrieves the certificate installed on the NCP.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	
EmberCertificateData localCert	The locally installed certificate.

Name: getCertificate283k1	**ID:** 0x00EC
Description: Retrieves the 283k certificate installed on the NCP.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	
EmberCertificate283k1Data localCert	The locally installed certificate.

Name: dsaSign	**ID:** 0x00A6
Description: LEGACY FUNCTION: This functionality has been replaced by a single bit in the EmberApsFrame, EMBER_APS_OPTION_DSA_SIGN. Devices wishing to send signed messages should use that as it requires fewer function calls and message buffering. The dsaSignHandler response is still called when EMBER_APS_OPTION_DSA_SIGN is used. However, this function is still supported. This function begins the process of signing the passed message contained within the messageContents array. If no other ECC operation is going on, it will immediately return with EMBER_OPERATION_IN_PROGRESS to indicate the start of ECC operation. It will delay a period of time to let APS retries take place, but then it will shut down the radio and consume the CPU processing until the signing is complete. This may take up to 1 second. The signed message will be returned in the dsaSignHandler response. Note that the last byte of the messageContents passed to this function has special significance. As the typical use case for DSA signing is to sign the ZCL payload of a DRLC Report Event Status message in SE 1.0, there is often both a signed portion (ZCL payload) and an unsigned portion (ZCL header). The last byte in the content of messageToSign is therefore used as a special indicator to signify how many bytes of leading data in the array should be excluded from consideration during the signing process. If the signature needs to cover the entire array (all bytes except last one), the caller should ensure that the last byte of messageContents is 0x00. When the signature operation is complete, this final byte will be replaced by the signature type indicator (0x01 for ECDSA signatures), and the actual signature will be appended to the original contents after this byte.	
Command Parameters:	
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The message contents for which to create a signature. Per above notes, this may include a leading portion of data not included in the signature, in which case the last byte of this array should be set to the index of the first byte to be considered for signing. Otherwise, the last byte of messageContents should be 0x00 to indicate that a signature should occur across the entire contents.
Response Parameters:	
EmberStatus status	EMBER_OPERATION_IN_PROGRESS if the stack has queued up the operation for execution. EMBER_INVALID_CALL if the operation can't be performed in this context, possibly because another ECC operation is pending.

Name: dsaSignHandler	**ID:** 0x00A7
Description: The handler that returns the results of the signing operation. On success, the signature will be appended to the original message (including the signature type indicator that replaced the startIndex field for the signing) and both are returned via this callback.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	The result of the DSA signing operation.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The message and attached which includes the original message and the appended signature.

Name: dsaVerify	**ID:** 0x00A3
Description: Verify that signature of the associated message digest was signed by the private key of the associated certificate.	
Command Parameters:	
EmberMessageDigest digest	The AES-MMO message digest of the signed data. If dsaSign command was used to generate the signature for this data, the final byte (replaced by signature type of 0x01) in the messageContents array passed to dsaSign is included in the hash context used for the digest calculation.
EmberCertificateData signerCertificate	The certificate of the signer. Note that the signer's certificate and the verifier's certificate must both be issued by the same Certificate Authority, so they should share the same CA Public Key.
EmberSignatureData receivedSig	The signature of the signed data.
Response Parameters:	
EmberStatus status	

Name: dsaVerifyHandler	**ID:** 0x0078
Description: This callback is executed by the stack when the DSA verification has completed and has a result. If the result is EMBER_SUCCESS, the signature is valid. If the result is EMBER_SIGNATURE_VERIFY_FAILURE then the signature is invalid. If the result is anything else then the signature verify operation failed and the validity is unknown.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	The result of the DSA verification operation.

Name: dsaVerify283k1	**ID:** 0x00B0
Description: Verify that signature of the associated message digest was signed by the private key of the associated certificate.	
Command Parameters:	
EmberMessageDigest digest	The AES-MMO message digest of the signed data. If dsaSign command was used to generate the signature for this data, the final byte (replaced by signature type of 0x01) in the messageContents array passed to dsaSign is included in the hash context used for the digest calculation.
EmberCertificate283k1Data signerCertificate	The certificate of the signer. Note that the signer's certificate and the verifier's certificate must both be issued by the same Certificate Authority, so they should share the same CA Public Key.
EmberSignature283k1Data receivedSig	The signature of the signed data.
Response Parameters:	
EmberStatus status	

Name: setPreinstalledCbkeData	**ID:** 0x00A2
Description: Sets the device's CA public key, local certificate, and static private key on the NCP associated with this node.	
Command Parameters:	
EmberPublicKeyData caPublic	The Certificate Authority's public key.
EmberCertificateData myCert	The node's new certificate signed by the CA.
EmberPrivateKeyData myKey	The node's new static private key.
Response Parameters:	
EmberStatus status	

Name: savePreinstalledCbkeData283k1	**ID:** 0x00ED
Description: Sets the device's 283k1 curve CA public key, local certificate, and static private key on the NCP associated with this node.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	

12 Mfglib Frames

Name: mfglibStart	**ID:** 0x0083
Description: Activate use of mfglib test routines and enables the radio receiver to report packets it receives to the mfgLibRxHandler() callback. These packets will not be passed up with a CRC failure. All other mfglib functions will return an error until the mfglibStart() has been called	
Command Parameters:	
bool rxCallback	true to generate a mfglibRxHandler callback when a packet is received.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: mfglibEnd	**ID:** 0x0084
Description: Deactivate use of mfglib test routines; restores the hardware to the state it was in prior to mfglibStart() and stops receiving packets started by mfglibStart() at the same time.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: mfglibStartTone	**ID:** 0x0085
Description: Starts transmitting an unmodulated tone on the currently set channel and power level. Upon successful return, the tone will be transmitting. To stop transmitting tone, application must call mfglibStopTone(), allowing it the flexibility to determine its own criteria for tone duration (time, event, etc.)	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: mfglibStopTone	**ID:** 0x0086
Description: Stops transmitting tone started by mfglibStartTone().	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: mfglibStartStream	**ID:** 0x0087
Description: Starts transmitting a random stream of characters. This is so that the radio modulation can be measured.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: mfglibStopStream	**ID:** 0x0088
Description: Stops transmitting a random stream of characters started by mfglibStartStream().	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: mfglibSendPacket	**ID:** 0x0089
Description: Sends a single packet consisting of the following bytes: packetLength, packetContents[0], ... , packetContents[packetLength - 3], CRC[0], CRC[1]. The total number of bytes sent is packetLength + 1. The radio replaces the last two bytes of packetContents[] with the 16-bit CRC for the packet.	
Command Parameters:	
uint8_t packetLength	The length of the packetContents parameter in bytes. Must be greater than 3 and less than 123.
uint8_t[] packetContents	The packet to send. The last two bytes will be replaced with the 16-bit CRC.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: mfglibSetChannel	**ID:** 0x008A
Description: Sets the radio channel. Calibration occurs if this is the first time the channel has been used.	
Command Parameters:	
uint8_t channel	The channel to switch to. Valid values are 11 to 26.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: mfglibGetChannel	**ID:** 0x008B
Description: Returns the current radio channel, as previously set via mfglibSetChannel().	
Command Parameters: None	
Response Parameters:	
uint8_t channel	The current channel.

Name: mfglibSetPower	**ID:** 0x008C
Description: First select the transmit power mode, and then include a method for selecting the radio transmit power. The valid power settings depend upon the specific radio in use. Ember radios have discrete power settings, and then requested power is rounded to a valid power setting; the actual power output is available to the caller via mfglibGetPower().	
Command Parameters:	
uint16_t txPowerMode	Power mode. Refer to txPowerModes in stack/include/ember-types.h for possible values.
int8_t power	Power in units of dBm. Refer to radio data sheet for valid range.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: mfglibGetPower	**ID:** 0x008D
Description: Returns the current radio power setting, as previously set via mfglibSetPower().	
Command Parameters: None	
Response Parameters:	
int8_t power	Power in units of dBm. Refer to radio data sheet for valid range.

Name: mfglibRxHandler	**ID:** 0x008E
Description: A callback indicating a packet with a valid CRC has been received.	
This frame is a response to the *callback* command.	
Response Parameters:	
uint8_t linkQuality	The link quality observed during the reception
int8_t rssi	The energy level (in units of dBm) observed during the reception.
uint8_t packetLength	The length of the packetContents parameter in bytes. Will be greater than 3 and less than 123.
uint8_t[] packetContents	The received packet (last 2 bytes are not FCS / CRC and may be discarded).

13 Bootloader Frames

Name: launchStandaloneBootloader	**ID:** 0x008F
Description: Quits the current application and launches the standalone bootloader (if installed) The function returns an error if the standalone bootloader is not present	
Command Parameters:	
uint8_t mode	Controls the mode in which the standalone bootloader will run. See the app. note for full details. Options are: STANDALONE_BOOTLOADER_NORMAL_MODE: Will listen for an over-the-air image transfer on the current channel with current power settings. STANDALONE_BOOTLOADER_RECOVERY_MODE: Will listen for an over-the-air image transfer on the default channel with default power settings. Both modes also allow an image transfer to begin with XMODEM over the serial protocol's Bootloader Frame.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: sendBootloadMessage	**ID:** 0x0090
Description: Transmits the given bootload message to a neighboring node using a specific 802.15.4 header that allows the EmberZNet stack as well as the bootloader to recognize the message, but will not interfere with other ZigBee stacks.	
Command Parameters:	
bool broadcast	If true, the destination address and pan id are both set to the broadcast address.
EmberEUI64 destEui64	The EUI64 of the target node. Ignored if the broadcast field is set to true.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The multicast message.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: getStandaloneBootloaderVersionPlatMicroPhy	**ID:** 0x0091
Description: Detects if the standalone bootloader is installed, and if so returns the installed version. If not return 0xffff. A returned version of 0x1234 would indicate version 1.2 build 34. Also return the node's version of PLAT, MICRO and PHY.	
Command Parameters: None	
Response Parameters:	
uint16_t bootloader_version	BOOTLOADER_INVALID_VERSION if the standalone bootloader is not present, or the version of the installed standalone bootloader.
uint8_t nodePlat	The value of PLAT on the node
uint8_t nodeMicro	The value of MICRO on the node
uint8_t nodePhy	The value of PHY on the node

Name: incomingBootloadMessageHandler	**ID:** 0x0092
Description: A callback invoked by the EmberZNet stack when a bootload message is received.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberEUI64 longId	The EUI64 of the sending node.
uint8_t lastHopLqi	The link quality from the node that last relayed the message.
int8_t lastHopRssi	The energy level (in units of dBm) observed during the reception.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The bootload message that was sent.

Name: bootloadTransmitCompleteHandler	**ID:** 0x0093
Description: A callback invoked by the EmberZNet stack when the MAC has finished transmitting a bootload message.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	An EmberStatus value of EMBER_SUCCESS if an ACK was received from the destination or EMBER_DELIVERY_FAILED if no ACK was received.
uint8_t messageLength	The length of the *messageContents* parameter in bytes.
uint8_t[] messageContents	The message that was sent.

Name: aesEncrypt	**ID:** 0x0094
Description: Perform AES encryption on plaintext using key.	
Command Parameters:	
uint8_t[16] plaintext	16 bytes of plaintext.
uint8_t[16] key	The 16-byte encryption key to use.
Response Parameters:	
uint8_t[16] ciphertext	16 bytes of ciphertext.

Name: overrideCurrentChannel	**ID:** 0x0095
Description: A bootloader method for selecting the radio channel. This routine only works for sending and receiving bootload packets. Does not correctly do ZigBee stack changes. NOTE: this API is not safe to call on multi-network devices and it will return failure when so. Use of the ember/ezspSetRadioChannel APIs are multi-network safe and are recommended instead.	
Command Parameters:	
uint8_t channel	The channel to switch to. Valid values are 11 to 26.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

14 ZLL Frames

Name: zllNetworkOps	**ID:** 0x00B2
Description: A consolidation of ZLL network operations with similar signatures; specifically, forming and joining networks or touch-linking.	
Command Parameters:	
EmberZllNetwork networkInfo	Information about the network.
EzspZllNetworkOperation op	Operation indicator.
int8_t radioTxPower	Radio transmission power.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: zllSetInitialSecurityState	**ID:** 0x00B3
Description: This call will cause the device to setup the security information used in its network. It must be called prior to forming, starting, or joining a network.	
Command Parameters:	
EmberKeyData networkKey	ZLL Network key.
EmberZllInitialSecurityState securityState	Initial security state of the network.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: zllSetSecurityStateWithoutKey	**ID:** 0x00CF
Description: This call will update ZLL security token information. Unlike emberZllSetInitialSecurityState, this can be called while a network is already established.	
Command Parameters:	
EmberZllInitialSecurityState securityState	Security state of the network.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: zllStartScan	**ID:** 0x00B4
Description: This call will initiate a ZLL network scan on all the specified channels.	
Command Parameters:	
uint32_t channelMask	The range of channels to scan.
int8_t radioPowerForScan	The radio output power used for the scan requests.
EmberNodeType nodeType	The node type of the local device.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: zllSetRxOnWhenIdle	**ID:** 0x00B5
Description: This call will change the mode of the radio so that the receiver is on for a specified amount of time when the device is idle.	
Command Parameters:	
uint32_t durationMs	The duration in milliseconds to leave the radio on.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: zllNetworkFoundHandler	**ID:** 0x00B6
Description: This call is fired when a ZLL network scan finds a ZLL network.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberZllNetwork networkInfo	Information about the network.
bool isDeviceInfoNull	Used to interpret deviceInfo field.
EmberZllDeviceInfoRecord deviceInfo	Device specific information.
uint8_t lastHopLqi	The link quality from the node that last relayed the message.
int8_t lastHopRssi	The energy level (in units of dBm) observed during reception.

Name: zllScanCompleteHandler	**ID:** 0x00B7
Description: This call is fired when a ZLL network scan is complete.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	Status of the operation.

Name: zllAddressAssignmentHandler	**ID:** 0x00B8
Description: This call is fired when network and group addresses are assigned to a remote mode in a network start or network join request.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberZllAddressAssignment addressInfo	Address assignment information.
uint8_t lastHopLqi	The link quality from the node that last relayed the message.
int8_t lastHopRssi	The energy level (in units of dBm) observed during reception.

Name: zllTouchLinkTargetHandler	**ID:** 0x00BB
Description: This call is fired when the device is a target of a touch link.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberZllNetwork networkInfo	Information about the network.

Name: zllGetTokens	**ID:** 0x00BC
Description: Get the ZLL tokens.	
Command Parameters: None	
Response Parameters:	
EmberTokTypeStackZllData data	Data token return value.
EmberTokTypeStackZllSecurity security	Security token return value.

Name: zllSetDataToken	**ID:** 0x00BD
Description: Set the ZLL data token.	
Command Parameters:	
EmberTokTypeStackZllData data	Data token to be set.
Response Parameters: None	

Name: zllSetNonZllNetwork	**ID:** 0x00BF
Description: Set the ZLL data token bitmask to reflect the ZLL network state.	
Command Parameters: None	
Response Parameters: None	

Name: isZllNetwork	**ID:** 0x00BE
Description: Is this a ZLL network?	
Command Parameters: None	
Response Parameters:	
bool isZllNetwork	ZLL network?

Name: zllSetRadioIdleMode	**ID:** 0x00D4
Description: This call sets the radio's default idle power mode.	
Command Parameters:	
EmberRadioPowerMode mode	The power mode to be set.
Response Parameters: None	

Name: setZllNodeType	**ID:** 0x00D5
Description: This call sets the default node type for a factory new ZLL device.	
Command Parameters:	
EmberNodeType nodeType	The node type to be set.
Response Parameters: None	

Name: setZllAdditionalState	**ID:** 0x00D6
Description: This call sets additional capability bits in the ZLL state.	
Command Parameters:	
uint16_t state	A mask with the bits to be set or cleared.
Response Parameters: None	

Name: zllOperationInProgress	**ID:** 0x00D7
Description: Is there a ZLL (Touchlink) operation in progress?	
Command Parameters: None	
Response Parameters:	
bool zllOperationInProgress	ZLL operation in progress?

Name: zllRxOnWhenIdleGetActive	**ID:** 0x00D8
Description: Is the ZLL radio on when idle mode is active?	
Command Parameters: None	
Response Parameters:	
bool zllRxOnWhenIdleGetActive	ZLL radio on when idle mode is active?

Name: getZllPrimaryChannelMask	**ID:** 0x00D9
Description: Get the primary ZLL (touchlink) channel mask.	
Command Parameters: None	
Response Parameters:	
uint32_t zllPrimaryChannelMask	The primary ZLL channel mask

Name: getZllSecondaryChannelMask	**ID:** 0x00DA
Description: Get the secondary ZLL (touchlink) channel mask.	
Command Parameters: None	
Response Parameters:	
uint32_t zllSecondaryChannelMask	The secondary ZLL channel mask

Name: setZllPrimaryChannelMask	**ID:** 0x00DB
Description: Set the primary ZLL (touchlink) channel mask	
Command Parameters:	
uint32_t zllPrimaryChannelMask	The primary ZLL channel mask
Response Parameters: None	

Name: setZllSecondaryChannelMask	**ID:** 0x00DC
Description: Set the secondary ZLL (touchlink) channel mask.	
Command Parameters:	
uint32_t zllSecondaryChannelMask	The secondary ZLL channel mask
Response Parameters: None	

Name: zllClearTokens	**ID:** 0x0025
Description: Clear ZLL stack tokens.	
Command Parameters: None	
Response Parameters: None	

15 WWAH Frames

Name: setParentClassificationEnabled	**ID:** 0x00E7
Description: Sets whether to use parent classification when processing beacons during a join or rejoin. Parent classification considers whether a received beacon indicates trust center connectivity and long uptime on the network	
Command Parameters:	
bool enabled	Enable or disable parent classification
Response Parameters: None	

Name: getParentClassificationEnabled	**ID:** 0x00F0
Description: Gets whether to use parent classification when processing beacons during a join or rejoin. Parent classification considers whether a received beacon indicates trust center connectivity and long uptime on the network	
Command Parameters: None	
Response Parameters:	
bool enabled	Enable or disable parent classification

Name: setLongUpTime	**ID:** 0x00E3
Description: sets the device uptime to be long or short	
Command Parameters:	
bool hasLongUpTime	if the uptime is long or not
Response Parameters: None	

Name: setHubConnectivity	**ID:** 0x00E4
Description: sets the hub connectivity to be true or false	
Command Parameters:	
bool connected	if the hub is connected or not
Response Parameters: None	

Name: isUpTimeLong	**ID:** 0x00E5
Description: checks if the device uptime is long or short	
Command Parameters: None	
Response Parameters:	
bool hasLongUpTime	if the uptime is long or not

Name: isHubConnected	**ID:** 0x00E6
Description: checks if the hub is connected or not	
Command Parameters: None	
Response Parameters:	
bool isHubConnected	if the hub is connected or not

16 Green Power Frames

Name: gpProxyTableProcessGpPairing	**ID:** 0x00C9
Description: Update the GP Proxy table based on a GP pairing.	
Command Parameters:	
uint32_t options	The options field of the GP Pairing command.
EmberGpAddress addr	The target GPD.
uint8_t commMode	The communication mode of the GP Sink.
uint16_t sinkNetworkAddress	The network address of the GP Sink.
uint16_t sinkGroupId	The group ID of the GP Sink.
uint16_t assignedAlias	The alias assigned to the GPD.
uint8_t[8] sinkIeeeAddress	The IEEE address of the GP Sink.
EmberKeyData gpdKey	The key to use for the target GPD.
uint32_t gpdSecurityFrameCounter	The GPD security frame counter.
uint8_t forwardingRadius	The forwarding radius.
Response Parameters:	
bool gpPairingAdded	Whether a GP Pairing has been created or not.

Name: dGpSend	**ID:** 0x00C6
Description: Adds/removes an entry from the GP Tx Queue.	
Command Parameters:	
bool action	The action to perform on the GP TX queue (true to add, false to remove).
bool useCca	Whether to use ClearChannelAssessment when transmitting the GPDF.
EmberGpAddress addr	The Address of the destination GPD.
uint8_t gpdCommandId	The GPD command ID to send.
uint8_t gpdAsduLength	The length of the GP command payload.
uint8_t[] gpdAsdu	The GP command payload.
uint8_t gpepHandle	The handle to refer to the GPDF.
uint16_t gpTxQueueEntryLifetimeMs	How long to keep the GPDF in the TX Queue.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: dGpSentHandler	**ID:** 0x00C7
Description: A callback to the GP endpoint to indicate the result of the GPDF transmission.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
uint8_t gpepHandle	The handle of the GPDF.

Name: gpepIncomingMessageHandler	**ID:** 0x00C5
Description: A callback invoked by the ZigBee GP stack when a GPDF is received.	
This frame is a response to the *callback* command.	
Response Parameters:	
EmberStatus status	The status of the GPDF receive.
uint8_t gpdLink	The gpdLink value of the received GPDF.
uint8_t sequenceNumber	The GPDF sequence number.
EmberGpAddress addr	The address of the source GPD.
EmberGpSecurityLevel gpdfSecurityLevel	The security level of the received GPDF.
EmberGpKeyType gpdfSecurityKeyType	The securityKeyType used to decrypt/authenticate the incoming GPDF.
bool autoCommissioning	Whether the incoming GPDF had the auto-commissioning bit set.
uint8_t bidirectionalInfo	Bidirectional information represented in bitfields, where bit0 holds the rxAfterTx of incoming gpdf and bit1 holds if tx queue is available for outgoing gpdf.
uint32_t gpdSecurityFrameCounter	The security frame counter of the incoming GDPF.
uint8_t gpdCommandId	The gpdCommandId of the incoming GPDF.
uint32_t mic	The received MIC of the GPDF.
uint8_t proxyTableIndex	The proxy table index of the corresponding proxy table entry to the incoming GPDF.
uint8_t gpdCommandPayloadLength	The length of the GPD command payload.
uint8_t[] gpdCommandPayload	The GPD command payload.

Name: gpProxyTableGetEntry	**ID:** 0x00C8
Description: Retrieves the proxy table entry stored at the passed index.	
Command Parameters:	
uint8_t proxyIndex	The index of the requested proxy table entry.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
EmberGpProxyTableEntry entry	An EmberGpProxyTableEntry struct containing a copy of the requested proxy entry.

Name: gpProxyTableLookup	**ID:** 0x00C0
Description: Finds the index of the passed address in the gp table.	
Command Parameters:	
EmberGpAddress addr	The address to search for
Response Parameters:	
uint8_t index	The index, or 0x00FF for not found

Name: gpSinkTableGetEntry	**ID:** 0x00DD
Description: Retrieves the sink table entry stored at the passed index.	
Command Parameters:	
uint8_t sinkIndex	The index of the requested sink table entry.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
EmberGpSinkTableEntry entry	An EmberGpSinkTableEntry struct containing a copy of the requested sink entry.

Name: gpSinkTableLookup	**ID:** 0x00DE
Description: Finds the index of the passed address in the gp table.	
Command Parameters:	
EmberGpAddress addr	The address to search for.
Response Parameters:	
uint8_t index	The index, or 0xFF for not found

Name: gpSinkTableSetEntry	**ID:** 0x00DF
Description: Retrieves the sink table entry stored at the passed index.	
Command Parameters:	
uint8_t sinkIndex	The index of the requested sink table entry.
EmberGpSinkTableEntry entry	An EmberGpSinkTableEntry struct containing a copy of the sink entry to be updated.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: gpSinkTableRemoveEntry	**ID:** 0x00E0
Description: Removes the sink table entry stored at the passed index.	
Command Parameters:	
uint8_t sinkIndex	The index of the requested sink table entry.
Response Parameters: None	

Name: gpSinkTableFindOrAllocateEntry	**ID:** 0x00E1
Description: Finds or allocates a sink entry	
Command Parameters:	
EmberGpAddress addr	An EmberGpAddress struct containing a copy of the gpd address to be found.
Response Parameters:	
uint8_t index	An index of found or allocated sink or 0xFF if failed.

Name: gpSinkTableClearAll	**ID:** 0x00E2
Description: Clear the entire sink table	
Command Parameters: None	
Response Parameters: None	

Name: gpSinkTableInit	**ID:** 0x0070
Description: Initializes Sink Table	
Command Parameters: None	
Response Parameters: None	

Name: gpSinkTableSetSecurityFrameCounter	**ID:** 0x00F5
Description: Sets security framecounter in the sink table	
Command Parameters:	
uint8_t index	Index to the Sink table
uint32_t sfc	Security Frame Counter
Response Parameters: None	

Name: gpSinkCommission	**ID:** 0x010A
Description: Puts the GPS in commissioning mode.	
Command Parameters:	
uint8_t options	commissioning options
uint16_t gpmAddrForSecurity	gpm address for security.
uint16_t gpmAddrForPairing	gpm address for pairing.
uint8_t sinkEndpoint	sink endpoint.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: gpTranslationTableClear	**ID:** 0x010B
Description: Clears all entries within the translation table.	
Command Parameters: None	
Response Parameters: None	

Name: gpSinkTableGetNumberOfActiveEntries	**ID:** 0x0118
Description: Return number of active entries in sink table.	
Command Parameters: None	
Response Parameters:	
Uint_t number_of_entries	Number of active entries in sink table.

17 Token Interface Frames

Name: getTokenCount	**ID:** 0x0100
Description: Gets the total number of tokens.	
Command Parameters: None	
Response Parameters:	
uint8_t count	Total number of tokens.

Name: getTokenInfo	**ID:** 0x0101
Description: Gets the token information for a single token at provided index	
Command Parameters:	
uint8_t index	Index of the token in the token table for which information is needed.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
EmberTokenInfo tokenInfo	Token information.

Name: getTokenData	**ID:** 0x0102
Description: Gets the token data for a single token with provided key	
Command Parameters:	
uint32_t token	Key of the token in the token table for which data is needed.
uint32_t index	Index in case of the indexed token.
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.
EmberTokenData tokenData	Token Data

Name: setTokenData	**ID:** 0x0103
Description: Sets the token data for a single token with provided key	
Command Parameters:	
uint32_t token	Key of the token in the token table for which data is to be set.
uint32_t index	Index in case of the indexed token.
EmberTokenData tokenData	Token Data
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: resetNode	**ID:** 0x0104
Description: Reset the node by calling halReboot.	
Command Parameters: None	
Response Parameters: None	

Name: gpSecurity Test Vectors	**ID:** 0x0117
Description: Run GP security test vectors.	
Command Parameters: None	
Response Parameters:	
EmberStatus status	An EmberStatus value indicating success or the reason for failure.

Name: tokenFactoryReset	**ID:** 0x0077
Description: Factory reset all configured Zigbee tokens.	
Command Parameters:	
bool excludeOutgoingFC	Exclude network and APS outgoing frame counter tokens.
Bool excludeBootCounter	Exclude stack boot counter token.
Response Parameters: None	

18 Alphabetical List of Frames

Name	ID
addEndpoint	0x0002
addressTableEntryIsActive	0x005B
aesEncrypt	0x0094
aesMmoHash	0x006F
bindingIsActive	0x002E
bootloadTransmitCompleteHandler	0x0093
broadcastNetworkKeySwitch	0x0074
broadcastNextNetworkKey	0x0073
calculateSmacs	0x009F
calculateSmacs283k1	0x00EA
calculateSmacsHandler	0x00A0
calculateSmacsHandler283k1	0x00EB
callback	0x0006
checkKeyContext	0x0110
childId	0x0106
childIndex	0x0107
childJoinHandler	0x0023
clearBindingTable	0x002A
clearKeyTable	0x00B1
clearStoredBeacons	0x003C
clearTemporaryDataMaybeStoreLinkKey	0x00A1
clearTemporaryDataMaybeStoreLinkKey283k1	0x00EE
clearTransientLinkKeys	0x006B
counterRolloverHandler	0x00F2
customFrame	0x0047
customFrameHandler	0x0054
dGpSend	0x00C6
dGpSentHandler	0x00C7
debugWrite	0x0012
delayTest	0x009D
deleteBinding	0x002D
dsaSign	0x00A6
dsaSignHandler	0x00A7
dsaVerify	0x00A3
dsaVerify283k1	0x00B0
dsaVerifyHandler	0x0078
dutyCycleHandler	0x004D
echo	0x0081

Name	ID
energyScanRequest	0x009C
energyScanResultHandler	0x0048
eraseKeyTableEntry	0x0076
exportKey	0x0114
exportLinkKeyByEui	0x010D
exportLinkKeyByIndex	0x010F
exportTransientKeyByEui	0x0113
exportTransientKeyByIndex	0x0112
findAndRejoinNetwork	0x0021
findKeyTableEntry	0x0075
findUnusedPanId	0x00D3
formNetwork	0x001E
generateCbkeKeys	0x00A4
generateCbkeKeys283k1	0x00E8
generateCbkeKeysHandler	0x009E
generateCbkeKeysHandler283k1	0x00E9
getAddressTableRemoteEui64	0x005E
getAddressTableRemoteNodeId	0x005F
getApsKeyInfo	0x010C
getBeaconClassificationParams	0x00F3
getBindingRemoteNodeId	0x002F
getCertificate	0x00A5
getCertificate283k1	0x00EC
setConcentrator	0x004A
getConfigurationValue	0x0052
getCtune	0x00F6
getCurrentDutyCycle	0x004C
getCurrentSecurityState	0x0069
getEui64	0x0026
getDutyCycleLimits	0x004B
getDutyCycleState	0x0035
getExtendedTimeout	0x007F
getExtendedValue	0x0003
getFirstBeacon	0x003D
getLibraryStatus	0x0001
getLogicalChannel	0x00BA
getMfgToken	0x000B
getMulticastTableEntry	0x0063
getNeighbor	0x0079

Name	ID
getNeighborFrameCounter	0x003E
getNetworkKeyInfo	0x0116
getNetworkParameters	0x0028
getNextBeacon	0x0004
getNodeId	0x0027
getNumStoredBeacons	0x0008
getParentChildParameters	0x0029
getParentClassificationEnabled	0x00F0
getPhyInterfaceCount	0x00FC
getPolicy	0x0056
getRadioParameters	0x00FD
getRandomNumber	0x0049
getRouteTableEntry	0x007B
getRoutingShortcutThreshold	0x00D1
getSourceRouteTableEntry	0x00C1
getSourceRouteTableFilledSize	0x00C2
getSourceRouteTableTotalSize	0x00C3
getStandaloneBootloaderVersionPlatMicroPhy	0x0091
getTimer	0x004E
getToken	0x000A
getTokenCount	0x0100
getTokenData	0x0102
getTokenInfo	0x0101
getTrueRandomEntropySource	0x004F
getValue	0x00AA
getXncpInfo	0x0013
getZllPrimaryChannelMask	0x00D9
getZllSecondaryChannelMask	0x00DA
gpProxyTableGetEntry	0x00C8
gpProxyTableLookup	0x00C0
gpProxyTableProcessGpPairing	0x00C9
gpSecurityTestVectors	0x0117
gpSinkCommission	0x010A
gpSinkTableClearAll	0x00E2
gpSinkTableFindOrAllocateEntry	0x00E1
gpSinkTableGetEntry	0x00DD
gpSinkTableGetNumberofActiveEntries	0x0118
gpSinkTableInit	0x0070
gpSinkTableLookup	0x00DE

Name	ID
gpSinkTableRemoveEntry	0x00E0
gpSinkTableSetEntry	0x00DF
gpTranslationTableClear	0x010B
gpepIncomingMessageHandler	0x00C5
idConflictHandler	0x007C
importKey	0x0115
importLinkKey	0x010E
importTransientKey	0x0111
incomingBootloadMessageHandler	0x0092
incomingManyToOneRouteRequestHandler	0x007D
incomingMessageHandler	0x0045
incomingRouteErrorHandler	0x0080
incomingNetworkStatusHandler	0x00C4
incomingRouteRecordHandler	0x0059
incomingSenderEui64Handler	0x0062
invalidCommand	0x0058
isHubConnected	0x00E6
isUpTimeLong	0x00E5
isZllNetwork	0x00BE
joinNetwork	0x001F
joinNetworkDirectly	0x003B
launchStandaloneBootloader	0x008F
leaveNetwork	0x0020
lookupEui64ByNodeId	0x0061
lookupNodeIdByEui64	0x0060
macFilterMatchMessageHandler	0x0046
macPassthroughMessageHandler	0x0097
maximumPayloadLength	0x0033
messageSentHandler	0x003F
mfglibEnd	0x0084
mfglibGetChannel	0x008B
mfglibGetPower	0x008D
mfglibRxHandler	0x008E
mfglibSendPacket	0x0089
mfglibSetChannel	0x008A
mfglibSetPower	0x008C
mfglibStart	0x0083
mfglibStartStream	0x0087
mfglibStartTone	0x0085

Name	ID
mfglibStopStream	0x0088
mfglibStopTone	0x0086
multiPhySetRadioChannel	0x00FB
multiPhySetRadioPower	0x00FA
multiPhyStart	0x00F8
multiPhyStop	0x00F9
neighborCount	0x007A
networkFoundHandler	0x001B
networkInit	0x0017
networkState	0x0018
noCallbacks	0x0007
nop	0x0005
overrideCurrentChannel	0x0095
permitJoining	0x0022
pollCompleteHandler	0x0043
pollForData	0x0042
pollHandler	0x0044
proxyBroadcaset	0x0037
rawTransmitCompleteHandler	0x0098
readAndClearCounters	0x0065
readAttribute	0x0108
readCounters	0x00F1
remoteDeleteBindingHandler	0x0032
remoteSetBindingHandler	0x0031
removeDevice	0x00A8
replaceAddressTableEntry	0x0082
requestLinkKey	0x0014
resetNode	0x0104
savePreinstalledCbkeData283k1	0x00ED
scanCompleteHandler	0x001C
sendBootloadMessage	0x0090
sendBroadcast	0x0036
sendLinkPowerDeltaRequest	0x00F7
sendManyToOneRouteRequest	0x0041
sendMulticast	0x0038
sendMulticastWithAlias	0x003A
sendRawMessage	0x0096
sendRawMessageExtended	0x0051

Name	ID
sendReply	0x0039
sendTrustCenterLinkKey	0x0067
sendUnicast	0x0034
setAddressTableRemoteEui64	0x005C
setAddressTableRemoteNodeId	0x005D
setBeaconClassificationParams	0x00EF
setBinding	0x002B
setBindingRemoteNodeId	0x0030
setBrokenRouteErrorCode	0x0011
setChildData	0x00AC
setConcentrator	0x0010
setConfigurationValue	0x0053
setCtune	0x00F5
setDutyCycleLimitsInStack	0x0040
setExtendedTimeout	0x007E
setGpioRadioPowerMask	0x00AE
setHubConnectivity	0x00E4
setInitialSecurityState	0x0068
setLogicalAndRadioChannel	0x00B9
setLongUpTime	0x00E3
setMacPollFailureWaitTime	0x00F4
setManufacturerCode	0x0015
setMulticastTableEntry	0x0064
setNeighborFrameCounter	0x00AD
setParentClassificationEnabled	0x00E7
setPassiveAckConfig	0x0105
setPolicy	0x0055
setPowerDescriptor	0x0016
setPreinstalledCbkeData	0x00A2
setPreinstalledCbkeData283k1	0x00ED
setRadioChannel	0x009A
setRadioIeee802154CcaMode	0x0095
setRadioPower	0x0099
setRoutingShortcutThreshold	0x00D0
setSourceRouteDiscoveryMode	0x005A
setTimer	0x000E
setToken	0x0009
setTokenData	0x0103
setValue	0x00AB

Name	ID
setZllAdditionalState	0x00D6
setZllNodeType	0x00D5
setZllPrimaryChannelMask	0x00DB
setZllSecondaryChannelMask	0x00DC
stackStatusHandler	0x0019
stackTokenChangeHandler	0x000D
startScan	0x001A
stopScan	0x001D
switchNetworkKeyHandler	0x006E
timerHandler	0x000F
tokenFactoryReset	0x0077
trustCenterJoinHandler	0x0024
unicastCurrentNetworkKey	0x0050
unicastNwkKeyUpdate	0x00A9
unusedPanIdFoundHandler	0x00D2
updateTcLinkKey	0x006C
version	0x0000
writeAttribute	0x0109
writeNodeData	0x00FE
zigbeeKeyEstablishmentHandler	0x009B
zllAddressAssignmentHandler	0x00B8
zllClearTokens	0x0025
zllGetTokens	0x00BC
zllNetworkFoundHandler	0x00B6
zllNetworkOps	0x00B2
zllOperationInProgress	0x00D7
zllRxOnWhenIdleGetActive	0x00D8
zllScanCompleteHandler	0x00B7
zllSetDataToken	0x00BD
zllSetInitialSecurityState	0x00B3
zllSetNonZllNetwork	0x00BF
zllSetRadioIdleMode	0x00D4
zllSetRxOnWhenIdle	0x00B5
zllSetSecurityStateWithoutKey	0x00CF
zllStartScan	0x00B4
zllTouchLinkTargetHandler	0x00BB

19 Numeric List of Frames

ID	Name
0x0000	version
0x0001	getLibraryStatus
0x0002	addEndpoint
0x0003	getExtendedValue
0x0004	getNextBeacon
0x0005	nop
0x0006	callback
0x0007	noCallbacks
0x0008	getNumStoredBeacons
0x0009	setToken
0x000A	getToken
0x000B	getMfgToken
0x000C	setMfgToken
0x000D	stackTokenChangedHandler
0x000E	setTimer
0x000F	timerHandler
0x0010	setConcentrator
0x0011	setBrokenRouteErrorCode
0x0012	debugWrite
0x0013	getXncpInfo
0x0014	requestLinkKey
0x0015	setManufacturerCode
0x0016	setPowerDescriptor
0x0017	networkInit
0x0018	networkState
0x0019	stackStatusHandler
0x001A	startScan
0x001B	networkFoundHandler
0x001C	scanCompleteHandler
0x001D	stopScan
0x001E	formNetwork
0x001F	joinNetwork
0x0020	leaveNetwork
0x0021	findAndRejoinNetwork
0x0022	permitJoining
0x0023	childJoinHandler
0x0024	trustCenterJoinHandler
0x0025	zllClearTokens

ID	Name
0x0026	getEui64
0x0027	getNodeId
0x0028	getNetworkParameters
0x0029	getParentChildParameters
0x002A	clearBindingTable
0x002B	setBinding
0x002C	getBinding
0x002D	deleteBinding
0x002E	bindingIsActive
0x002F	getBindingRemoteNodeId
0x0030	setBindingRemoteNodeId
0x0031	remoteSetBindingHandler
0x0032	remoteDeleteBindingHandler
0x0033	maximumPayloadLength
0x0034	sendUnicast
0x0035	getDutyCycleState
0x0036	sendBroadcast
0x0037	proxyBroadcast
0x0038	sendMulticast
0x0039	sendReply
0x003A	sendMulticastWithAlias
0x003B	joinNetworkDirectly
0x003C	clearStoredBeacons
0x003D	getFirstBeacon
0x003E	getNeighborFrameCounter
0x003F	messageSentHandler
0x0040	setDutyCycleLimitsInStack
0x0041	sendManyToOneRouteRequest
0x0042	pollForData
0x0043	pollCompleteHandler
0x0044	pollHandler
0x0045	incomingMessageHandler
0x0046	macFilterMatchMessageHandler
0x0047	customFrame
0x0048	energyScanResultHandler
0x0049	getRandomNumber
0x004A	getChildData
0x004B	getDutyCycleLimits
0x004C	getCurrentDutyCycle

ID	Name
0x004D	dutyCycleHandler
0x004E	getTimer
0x004F	getTrueRandomEntropySource
0x0050	unicastCurrentNetworkKey
0x0051	sendRawMessageExtended
0x0052	getConfigurationValue
0x0053	setConfigurationValue
0x0054	customFrameHandler
0x0055	setPolicy
0x0056	getPolicy
0x0057	--unassigned--
0x0058	invalidCommand
0x0059	incomingRouteRecordHandler
0x005A	setSourceRouteDiscoveryMode
0x005B	addressTableEntryIsActive
0x005C	setAddressTableRemoteEui64
0x005D	setAddressTableRemoteNodeId
0x005E	getAddressTableRemoteEui64
0x005F	getAddressTableRemoteNodeId
0x0060	lookupNodeIdByEui64
0x0061	lookupEui64ByNodeId
0x0062	incomingSenderEui64Handler
0x0063	getMulticastTableEntry
0x0064	setMulticastTableEntry
0x0065	readAndClearCounters
0x0066	– unassigned --
0x0067	sendTrustCenterLinkKey
0x0068	setInitialSecurityState
0x0069	getCurrentSecurityState
0x006A	– unassigned --
0x006B	clearTransientLinkKeys
0x006C	updateTcLinkKey
0x006D	– unassigned --
0x006E	switchNetworkKeyHandler
0x006F	aesMmoHash
0x0070	gpSinkTableInit
0x0071	—unassigned --
0x0072	– unassigned --
0x0073	broadcastNextNetworkKey

ID	Name
0x0074	broadcastNetworkKeySwitch
0x0075	findKeyTableEntry
0x0076	eraseKeyTableEntry
0x0077	tokenFactoryReset
0x0078	dsaVerifyHandler
0x0079	getNeighbor
0x007A	neighborCount
0x007B	getRouteTableEntry
0x007C	idConflictHandler
0x007D	incomingManyToOneRouteRequestHandler
0x007E	setExtendedTimeout
0x007F	getExtendedTimeout
0x0080	incomingRouteErrorHandler
0x0081	echo
0x0082	replaceAddressTableEntry
0x0083	mfglibStart
0x0084	mfglibEnd
0x0085	mfglibStartTone
0x0086	mfglibStopTone
0x0087	mfglibStartStream
0x0088	mfglibStopStream
0x0089	mfglibSendPacket
0x008A	mfglibSetChannel
0x008B	mfglibGetChannel
0x008C	mfglibSetPower
0x008D	mfglibGetPower
0x008E	mfglibRxHandler
0x008F	launchStandaloneBootloader
0x0090	sendBootloadMessage
0x0091	getStandaloneBootloaderVersionPlatMicroPhy
0x0092	incomingBootloadMessageHandler
0x0093	bootloadTransmitCompleteHandler
0x0094	aesEncrypt
0x0095	setRadioIeee802154CcaMode
0x0096	sendRawMessage
0x0097	macPassthroughMessageHandler
0x0098	rawTransmitCompleteHandler
0x0099	setRadioPower
0x009A	setRadioChannel

ID	Name
0x009B	zigbeeKeyEstablishmentHandler
0x009C	energyScanRequest
0x009D	delayTest
0x009E	generateCbkeKeysHandler
0x009F	calculateSmacs
0x00A0	calculateSmacsHandler
0x00A1	clearTemporaryDataMaybeStoreLinkKey
0x00A2	setPreinstalledCbkeData
0x00A3	dsaVerify
0x00A4	generateCbkeKeys
0x00A5	getCertificate
0x00A6	dsaSign
0x00A7	dsaSignHandler
0x00A8	removeDevice
0x00A9	unicastNwkKeyUpdate
0x00AA	getValue
0x00AB	setValue
0x00AC	setChildData
0x00AD	setNeighborFrameCounter
0x00AE	setGpioRadioPowerMask
0x00AF	– unassigned --
0x00B0	dsaVerify283k1
0x00B1	clearKeyTable
0x00B2	zllNetworkOps
0x00B3	zllSetInitialSecurityState
0x00B4	zllStartScan
0x00B5	zllSetRxOnWhenIdle
0x00B6	zllNetworkFoundHandler
0x00B7	zllScanCompleteHandler
0x00B8	zllAddressAssignmentHandler
0x00B9	setLogicalAndRadioChannel
0x00BA	getLogicalChannel
0x00BB	zllTouchLinkTargetHandler
0x00BC	zllGetTokens
0x00BD	zllSetDataToken
0x00BE	isZllNetwork
0x00BF	zllSetNonZllNetwork
0x00C0	gpProxyTableLookup
0x00C1	getSourceRouteTableEntry

ID	Name
0x00C2	getSourceRouteTableFilledSize
0x00C3	getSourceRouteTableTotalSize
0x00C4	incomingNetworkStatusHandler
0x00C5	gpepIncomingMessageHandler
0x00C6	dGpSend
0x00C7	dGpSentHandler
0x00C8	gpProxyTableGetEntry
0x00C9	gpProxyTableProcessGpPairing
0x00CA	– unassigned --
0x00CB	– unassigned --
0x00CC	– unassigned --
0x00CD	– unassigned --
0x00CE	– unassigned --
0x00CF	zllSetSecurityStateWithoutKey
0x00D0	setRoutingShortcutThreshold
0x00D1	getRoutingShortcutThreshold
0x00D2	unusedPanIdFoundHandler
0x00D3	findUnusedPanId
0x00D4	zllSetRadioIdleMode
0x00D5	setZllNodeType
0x00D6	setZllAdditionalState
0x00D7	zllOperationInProgress
0x00D8	zllRxOnWhenIdleGetActive
0x00D9	getZllPrimaryChannelMask
0x00DA	getZllSecondaryChannelMask
0x00DB	setZllPrimaryChannelMask
0x00DC	setZllSecondaryChannelMask
0x00DD	gpSinkTableGetEntry
0x00DE	gpSinkTableLookup
0x00DF	gpSinkTableSetEntry
0x00E0	gpSinkTableRemoveEntry
0x00E1	gpSinkTableFindOrAllocateEntry
0x00E2	gpSinkTableClearAll
0x00E3	setLongUpTime
0x00E4	setHubConnectivity
0x00E5	isUpTimeLong
0x00E6	isHubConnected
0x00E7	setParentClassificationEnabled
0x00E8	generateCbkeKeys283k1

ID	Name
0x00E9	generateCbkeKeysHandler283k1
0x00EA	calculateSmacs283k1
0x00EB	calculateSmacsHandler283k1
0x00EC	getCertificate283k1
0x00ED	savePreinstalledCbkeData283k1
0x00EE	clearTemporaryDataMaybeStoreLinkKey283k1
0x00EF	setBeaconClassificationParams
0x00F0	getParentClassificationEnabled
0x00F1	readCounters
0x00F2	counterRolloverHandler
0x00F3	getBeaconClassificationParams
0x00F4	setMacPollFailureWaitTime
0x00F5	--unassigned--
0x00F6	--unassigned--
0x00F7	sendLinkPowerDeltaRequest
0x00F8	multiPhyStart
0x00F9	multiPhyStop
0x00FA	multiPhySetRadioPower
0x00FB	multiPhySetRadioChannel
0x00FC	getPhyInterfaceCount
0x00FD	getRadioParameters
0x00FE	writeNodeData
0x00FF	getRadioChannel
0x0100	getTokenCount
0x0101	getTokenInfo
0x0102	getTokenData
0x0103	setTokenData
0x0104	resetNode
0x0105	setPassiveAckConfig
0x0106	childId
0x0107	childIndex
0x0108	readAttribute
0x0109	writeAttribute
0x010A	gpSinkCommission
0x010B	gpTranslationTableClear
0x010C	getApsKeyInfo
0x010D	exportLinkKeyByEui
0x010E	importLinkKey
0x010F	exportLinkKeyByIndex

ID	Name
0x0110	checkKeyContext
0x0111	importTransientKey
0x0112	exportTransientKeyByIndex
0x0113	exportTransientKeyByEui
0x0114	exportKey
0x0115	importKey
0x0116	getNetworkKeyInfo
0x0117	gpSecurityTestVectors
0x0118	gpSinkTableGetNumberofActiveEntries