Z-Stack Linux Gateway

Application Programming Interface document

Version: 1.1

Created May 1, 2014

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

The purpose of this document is to provide a description of the application programming interface between the Z-Stack Linux Gateway solution provided by Texas Instruments for Home Automation gateways and the applications that intend to use it across that interface.

The Z-Stack Linux Gateway solution provided by Texas Instruments implements functionalities that allow managing a network of ZigBee devices, taking care of the procedures associated to the ZigBee protocol, and providing a high level data pipe for end user applications, hence abstracting the ZigBee inner workings for the application developer, thus making the integration of a ZigBee technology simpler.

The API described here illustrates functionality, name and set of parameters that must be configured or parsed by an application that intend to use that API across the interface and make use of the ZigBee gateway software.

This API is defined at a specific interface level, which is a TCP socket pipe.

The ZigBee gateway software supports one client application active at a specific time and connected to the network manager /gateway server socket service access point interface.

API commands and parameters are serialized over the socket interface using Google protocol buffers (protobuf; for details see https://developers.google.com/protocol-buffers/). Please note that the name of some parameters in this API document does not reflect the true name of the parameter which will be used by the application using that API, as those parameters are automatically generated by the protobuf engine).

For transport using a TCP socket, the protobuf-packed packets are preceded by a 4 bytes header, which is described in **Appendix B – API Packet Header**.

2 Model and Conventions

The model and naming conventions used to define and describe the API have been taken from the Open System Interconnect (OSI) model, which is more apt to define interfaces for layers that implement protocol stack functionalities.

According to this model, the layer below the API interface level (the service provider, i.e. the ZigBee gateway SW solution in this case) provides services to the layer above (the gateway application).

The API, following the OSI model, defines clear roles and interfacing mechanics between layer N and N+1, in the form of service calls: Request/Confirm and Indication/Response.

Layer N+1 or service user, in this document referred to as the gateway application, makes API *Request* and *Responses* API calls.

Layer N or service provider, in this document, refers the gateway subsystem, i.e. may confirm the request when needed with a corresponding *Confirm* API call. When an un-solicited message or a notification needs to be sent to the gateway application (as an effect of an incoming over-the-air (OTA) message being received or as a result of local operation), the gateway subsystem makes an *Indication* API call, which may be responded by the gateway application with the corresponding Response API.

3 System block diagram

The main system blocks components of the gateway system are illustrated in Figure 1.
The gateway subsystem includes all system components below the ‘gateway application’.
As indicated below, the HA gateway server and the Network manager server have separate TCP socket connections and service access point interfaces (SAPI).

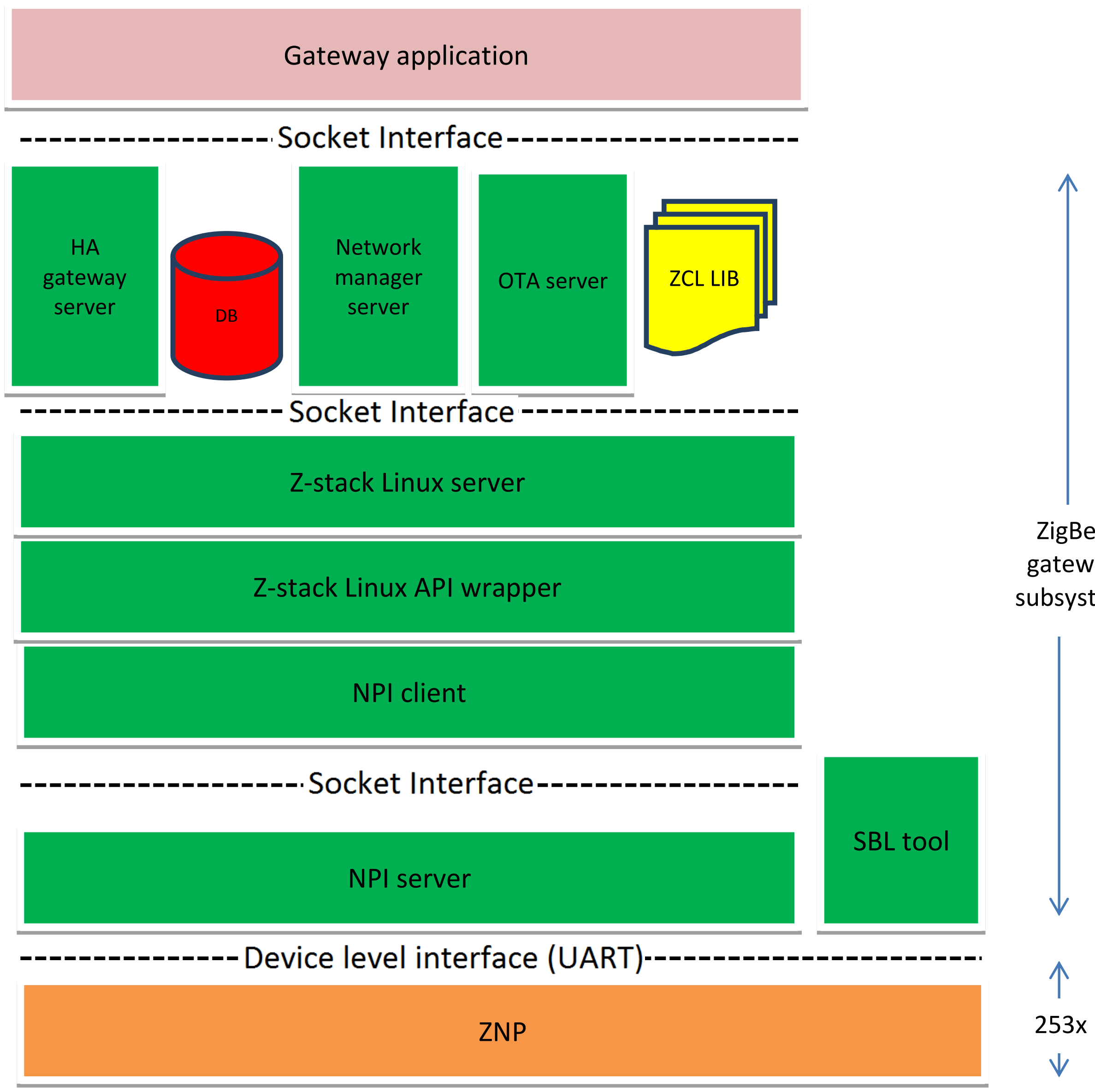

Figure 1: Gateway block diagram

4 API Command Types

In this paragraph, the following terms are being used:

- **Outgoing**
  Generally refers to commands sent from the gateway application to the gateway subsystem.

- **Incoming**
  Generally refers to commands sent from the gateway subsystem to the gateway application.

- **Outgoing request**
  An outgoing API command.
  Each outgoing request is immediately followed by a confirmation message in a synchronous manner. The gateway application shall not send a new outgoing request until it receives the confirmation for the previous request.

- **Confirmation message**
  An incoming API command that confirms the status of an outgoing request.

  - **Specific confirmation**
    A confirmation message that carries the information that was requested by the outgoing request.
    Specific confirmations are never followed by incoming responses.

  - **Generic confirmation**
    - **without sequence number**
      A confirmation message that does not carry command-specific information.
    - **with sequence number**
      A confirmation message for commands that generate incoming responses.
      Specifies the sequence number that will be carried by the resulting incoming responses.
      If the status is other than success, no sequence number is specified, and no response follows.

- **Incoming response**
  An incoming API message that is received as a response for an outgoing command.
  It is triggered by a received over-the-air message.
  It can arrive at any time after the respective outgoing request and the related confirmation message, and carries the same sequence number as specified in the confirmation. The sequence number provides a way for an incoming response to be related to the respective outgoing request.
  Since it depend on over-the-air communication, it is possible that an outgoing request will never get its incoming responses (e.g. when the device being contacted is turned off before being able to respond). For that reason, it is recommended that the application implement a timeout when waiting for responses.

- **Specific response**
  An incoming response that carries information that is specific to the respective outgoing request. It is possible that a single outgoing request will result in multiple specific responses (e.g. when an outgoing request was sent as groupcast or broadcast).

- **Generic response**
  An incoming response that does not carry specific information, other than the status of the request as reported by the responding device. Generic responses are only used for unicast outgoing requests. Outgoing requests that are sent as groupcast or broadcast will not be followed by Generic responses.

The various API commands described in this document are defined by the following command types (Flow diagrams for each of the API command types are available at **Appendix A - API command type diagrams**):

- **Incoming Indication**
  - Incoming API commands that are mostly triggered by over-the-air ZigBee events, and are not directly related to an outgoing request.
  - Does not require a response from the application.
- **Incoming Request**
  - Incoming API commands that are mostly triggered by over-the-air requests from remote ZigBee devices.
  - The application is expected to send an outgoing response to the device that sent the request.
- **Outgoing response**
  - Outgoing API command that is sent by the application as a response to an Incoming Request.
  - The sequence number of the outgoing response shall be copied from the respective incoming request.
  - Not followed by confirmation or incoming response messages.
- **No-response command**
  - ***Outgoing requests*** which are either processed internally in the gateway subsystem, or are triggering over the air activity that does not require returning communication.
  - Followed by either a ***specific confirmation***, in case data should be returned to the application, or ***generic confirmation without sequence number*** otherwise. In a single case (**NWK_ZIGBEE_SYSTEM_SELF_SHUTDOWN_REQ**) there is no confirmation at all.
- **Specific-response command**
  - ***Outgoing requests*** which (unless addressed to self) trigger over-the-air activity that are expected to result in command-specific response(s) from remote device(s).
  - If not addressed to self:
    - Followed by a ***generic confirmation with sequence number***.
    - When responses from the remote devices are received, respective ***Specific response*** commands are sent to the application.

  - If addressed to self:
    - No confirmation is sent.
    - A ***Specific response*** is immediately sent to the application (in a synchronous manner), with the sequence number set to 0xFFFF (which means that this response refers to the request that immediately precedes it).
- **Generic-response command**
  - ***Outgoing requests*** which trigger over-the-air activity that may result in a generic response from a remote device.
  - When such a command is sent by the application as a unicast, the gateway immediately responds with a ***generic confirmation with sequence number***. Then, a ***Generic response*** will follow, when the response is received from the destination device.
  - When such a command is sent by the application as a groupcast or broadcast, the gateway immediately responds with a ***generic confirmation without sequence number***. No response will follow.

The use of sequence numbers allows the application to keep track of request-response transactions, and provides support for multiple pending transactions of the same type.

Note: Incoming Indications and incoming requests may be sent by the gateway subsystem asynchronously: it is possible that the application sends an outgoing request, and then it receives an unrelated incoming indication before it receives the confirmation message for the outgoing request.

5 Addressing

In this API document, addressing parameters are specified for almost every command. For simplicity, the following address structure is defined, which is a superset of the possible addressing fields. In the rest of the document, this structure is used instead of specifying the specific address fields each time.

Structure type name: **AddressStruct_t**

Field Name	Type	When should be specified	Values
AddressType	Uint8	Always	0 – Unicast / Single device (using 64bit IEEE address) 1 – Groupcast (Using 16bit Group ID) 2 – Broadcast (Using 16bit broadcast address) 3 – Self addressing – referring to the gateway device itself. No address is specified.
IEEE Address	Uint64	AddressType == Unicast	Device's unique 64bit IEEE address.
Group Address	Uint16	AddressType == Groupcast	16bit group address
Broadcast Address	Uint16	AddressType == Broadcast	16bit broadcast address
EndpointID	Uint8	Optional, as specified by each specific command (see below).	Endpoint ID. If omitted, it means referring to all endpoint of the device (unless irrelevant)

When this structure is referenced, it's type is always preceded by a list of the allowed addressing modes: U – Unicast, G – Groupcast, B – Broadcast, S – Self, E – end point should be specified, e – end point is optional.

For example, ***AddressStruct_t {U/G,e}*** means that for the specific command, Unicast or Groupcast must be selected, and that the EndpointID can be specified, but does not have to.

6 API Grouping

The API commands are grouped in chapters by the module that is in charge of their initial processing: Network manager server, Gateway sybsystem server, OTA server.
They are also further grouped by functionality, hinted by the first word in each API name:

- **ZIGBEE_** - Generic commands that are used by multiple servers.
- **NWK_** - commands for controlling the network structure and the behavior of the gateway device in the network
- **GW_** - commands for generic interaction with ZigBee devices in the network
- **DEV_** - perform device-specific interaction with ZigBee devices on the network
- **OTA_** - Over-The-Air upgrade procedures

7 Generic API set

These APIs can be sent/received (according to the command type) by any of the servers (Gateway, Network Manager, OTA). They are not specific to a certain server.

7.1 Generic confirmations and responses

7.1.1 ZIGBEE_GENERIC_CNF

Command type: **Generic confirmation**

This API is called by the gateway subsystem to notify the application about the status of a command that was just sent by the application. It is sent synchronously, immediately following the command which it confirms.

7.1.1.1 Parameter list

Parameter	***Type***	***Description***
Status	Uint16	see **Status and Error Codes**
SequenceNumber	Uint16	Optional. See **API Command Types**

7.1.2 ZIGBEE_GENERIC_RSP_IND

Command type: **Generic response**

This API is called by the gateway subsystem to notify the application about the status of a command that was previously sent by the application. It is sent asynchronously, when a default response is received over-the-air from the destination device.

7.1.2.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Same number that was specified in the respective confirmation message.
Status	Uint16	see **Status and Error Codes**

8 Network manager server API set

The network manager server API set implements the housekeeping functions of the application; in fact it is used to instantiate procedures that are related to manage/access ZigBee devices in the network, from a ‘ZigBee device protocol’ standpoint. For instance: notifying the application when a ZigBee device has joined the network, understanding which ZigBee devices are currently in the network and what services they support, maintaining routes and links between the nodes, removing devices from the network, arbitrating pairings between applications on remote devices. The network manager server API set is instantiated over the TCP socket service access point that interfaces with the network manager server SW entity.

Calling network manager server API may not necessarily result in an over the air message being generated or received by the gateway node.

8.1 Local device control/information API set

8.1.1 NWK_ZIGBEE_SYSTEM_RESET_REQ

Command type: **No-response command** / **NWK_ZIGBEE_SYSTEM_RESET_CNF**

This API is called by the gateway application to reset the ZigBee subsystem. It supports two reset modes: One mode is soft reset, where the ZigBee stack database is maintained (i.e. the gateway node re-appears in the ZigBee network using its previous configuration, maintaining all knowledge about the other devices in the network). The other mode is hard reset, AKA reset to factory new, where the entire network information is wiped out. The devices that were part of the network before a hard reset will not be able to communicate with the gateway unless they are (manually) joined again.

With both reset modes, all underlying state machines in the gateway subsystem (network manager server, gateway server, etc.) are reset, which discards out all outstanding messages that are still being processed. The gateway application should clear its own state machines accordingly.

The application shall expect to lose the socket connections to the servers right after it issues a NWK_ZIGBEE_SYSTEM_RESET_REQ. The application shall keep trying to reconnect to the servers as soon as it detects a disconnection. The reset confirmation will be sent to the application after it successfully reconnects to the network manager server. Since the reset confirmation is sent after the actual reset happens, it may take up to a few seconds for the confirmation to be received following a reset request.

In addition to the reset confirmation, when the ZigBee subsystem has restored its operation, it will report the network information using the **NWK_ZIGBEE_NWK_READY_IND**, just like it does following power up.

8.1.1.1 Parameter list

Parameter	***Type***	***Description***
ResetMode	Uint8	0 – Soft Reset / Maintain core stack base 1 – Hard Reset / Reset to factory new

8.1.2 NWK_ZIGBEE_SYSTEM_RESET_CNF

Command type: **Specific confirmation**

This API is called by the gateway subsystem to report that it has successfully completed a reset request.

8.1.2.1 Parameter list

Parameter	***Type***	***Description***
Status	Uint8	see **Status and Error Codes**
ResetMode	Uint8	As defined in the respective request

8.1.3 NWK_ZIGBEE_SYSTEM_SELF_SHUTDOWN_REQ
Command type: **No-response command** / no confirmation

This API is a trigger command called by the gateway application to shut down the ZigBee subsystem. The application shall expect to lose the connections to the ZigBee subsystem servers as soon as it issues this command.

No confirmation is sent to the application following this command. The application may monitor the gateway sybsystem processes to verify they have been terminated as expected.

After this command is issued, restarting the gateway shall be done by starting the gateway subsystem executables as usual (e.g. via a shell script).

8.1.3.1 Parameter list

(This command has no parameters)

8.1.4 NWK_SET_ZIGBEE_POWER_MODE_REQ

Command type: **No-response command** / **NWK_SET_ZIGBEE_POWER_MODE_CNF**

This API is called by the gateway application to control the ZigBee device power mode state. It is intended for shutting down the ZigBee SOC when ZigBee operation is not required, with the ability to turn it on again using a simple API command (in contrast to the NWK_ZIGBEE_SYSTEM_SELF_SHUTDOWN_REQ command, where turning the ZigBee subsystem back on must be done externally).

When the low power mode is selected, the gateway subsystem will turn off the ZigBee device, and send back a confirmation to the application. Then, when the active mode is selected, the gateway subsystem will turn the device back on and reset the complete ZigBee subsystem (soft reset), before sending the wake-up confirmation to the application. This means that the application shall treat resuming from sleep like it treats restarting following a soft reset request.

In addition to the wake-up confirmation, when the ZigBee subsystem has restored its operation, it will report the network information using the **NWK_ZIGBEE_NWK_READY_IND**, just like it does following power up.

Note that after sending each of the sleep request and the wake-up request, the application shall expect to lose connection with the gateway subsystem's server. Following a sleep request, the application shall continuously try to reconnect to the network manager server. Following a wake-up request, the applicatuion shall continuously try to reconnect to both the network manager and the gateway servers. (The gateway server is not available when the gateway subsystem is in sleep mode).

8.1.4.1 Parameter list

Parameter	***Type***	***Description***
PowerMode	Uint8	0- Sleep / Low Power mode 1- Wakeup / Active mode

8.1.5 NWK_SET_ZIGBEE_POWER_MODE_CNF

Command type: **Specific confirmation**

This API is called by the gateway subsystem to report that it has successfully completed a power mode request.

8.1.5.1 Parameter list

Parameter	***Type***	***Description***
Status	Uint8	see **Status and Error Codes**
PowerMode	Uint8	As defined in the respective request

8.1.6 NWK_GET_LOCAL_DEVICE_INFO_REQ

Command type: **No-response command** / **NWK_GET_LOCAL_DEVICE_INFO_CNF**

This API is called by the gateway application to retrieve information regarding the gateway device itself (e.g. list of registered endpoints and their descriptors, which is populated and configured by the gateway subsystem through a configuration file at startup).

8.1.6.1 Parameter list

(This command has no parameters)

8.1.7 NWK_GET_LOCAL_DEVICE_INFO_CNF

Command type: **Specific confirmation**

This API is called by the gateway subsystem to report the information relative to the gateway device.

8.1.7.1 Parameter list

Parameter	***Type***	***Description***
DeviceInfo	DeviceInfo_t	device information

Where DeviceInfo_t is defined as in **8.3.1.1**.

8.2 Network control/information API set

8.2.1 NWK_ZIGBEE_NWK_READY_IND

Command type: **Incoming Indication**

This API reports information about the current ZigBee network, where the gateway sits in. When an earlier GATEWAY_ZIGBEE_RESET_REQ with ‘Reset to factory new’ parameter has been issued, or after first power on when the device was factory new, the gateway subsystem reports this information after the network has been formed. If instead the parameter of the was ‘Maintain core stack base’ or a **NWK_SET_ZIGBEE_POWER_MODE_REQ** (with active mode) was requested, then this indication is sent to the application to inform it that the normal stack operation has been resumed

8.2.1.1 Parameter list

Parameter	***Type***	***Description***
NetworkChannel	Uint8	MAC channel ID
PanID	Uint16	Network PAN ID
ExtPanID	Uint64	Extended PAN ID

8.2.2 NWK_ZIGBEE_NWK_INFO_REQ
Command type: **No-response command** / **NWK_ZIGBEE_NWK_INFO_CNF**

This API is called by the gateway application to request information about the current ZigBee network.

8.2.2.1 *Parameter list*

(This command has no parameters)

8.2.3 NWK_ZIGBEE_NWK_INFO_CNF
Command type: **Specific confirmation**

This API called by the gateway subsystem to report to the gateway application information about the current ZigBee network.

8.2.3.1 *Parameter list*

Parameter	***Type***	***Description***
Status	Uint8	0 – network down 1 – network up
NetworkChannel	Uint8	MAC channel ID
PanID	Uint16	Network PAN ID
ExtPanID	Uint64	Extended PAN ID

8.2.4 NWK_SET_PERMIT_JOIN_REQ

Command type: **No-response command** / **ZIGBEE_GENERIC_CNF**

This API is called by the gateway application to set the open the network for joining

8.2.4.1 Parameter list

Parameter	***Type***	***Description***
PermitJoinType	Uint8	0 – permit join flag only set locally, on the gateway device itself 1 – permit join flag set for all devices in the network (including the gateway)
PermitJoinTime	Uint8	Time in seconds during which association is allowed (permit join remains true) 0 – turn permit join off (cancel) 0xff - unlimited time (permit until cancelled) Any other value – permit join time in seconds

8.2.5 NWK_MANAGE_PERIODIC_MTO_ROUTE_REQ

Command type: **No-response command** / **ZIGBEE_GENERIC_CNF**

This API is called by the gateway application to start a periodic many-to-one route request maintenance scheme or stop it. By default, the gateway subsystem enables MTO scheme which is configured with default parameters in a configuration file.

8.2.5.1 Parameter list

Parameter	***Type***	***Description***
Mode	Uint8	0 – start 1 – stop

8.2.6 NWK_GET_NEIGHBOR_TABLE_REQ

Command type: **Specific-response command** // **NWK_GET_NEIGHBOR_TABLE_RSP_IND**

This API is called by the gateway application when it wants to retrieve the neighbor table of a destination device or of the gateway device itself.

8.2.6.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U, S}	Note: This command cannot be sent to ZigBee End Devices.
StartIndex	Uint8	Specifies where to start in the response array list. The result may contain more entries than can be reported, so this field allows the user to retrieve the responses anywhere in the array list.

8.2.7 NWK_GET_NEIGHBOR_TABLE_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem as a response to **NWK_GET_NEIGHBOR_TABLE_REQ**.

8.2.7.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U}	The responder's address
NeighborTableEntries	Uint8	Total number of entries available in the device.
StartIndex	Uint8	Where in the total number of entries this response starts.
NeighborList	NeighborInfo_t []	Array. number of elements <= NeighborTableEntries - StartIndex

Where NeighborInfo_t is defined as follows:

Parameter	***Type***	***Description***
ExtendedPanId	Uint64	Extended PAN ID of the neighbor device
ExtendedAddress	Uint64	IEEE address
NetworkAddress	Uint16	Short address
DeviceType/ RxOnWhenIdle/ Relationship	Uint8	DeviceType: bits 1-0 RxOnWhenIdle: bits 3-2 Relationship: bits 6-4
PermitJoining	Uint8	PermitJoining: bits 1-0
Depth	Uint8	
LQI	Uint8	

8.2.8 NWK_GET_ROUTING_TABLE_REQ
Command type: **Specific-response command** // **NWK_GET_ROUTING_TABLE_RSP_IND**

This API is called by the gateway application when it wants to retrieve the routing table of a destination device.

8.2.8.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U}	
StartIndex	Uint8	Specifies where to start in the response array list. The result may contain more entries than can be reported, so this field allows the user to retrieve the responses anywhere in the array list.

8.2.9 NWK_GET_ROUTING_TABLE_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem as a response to **NWK_GET_ROUTING_TABLE_REQ**.

8.2.9.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U}	The responder's address
RoutingTableEntries	Uint8	Total number of entries available in the device.
StartIndex	Uint8	Where in the total number of entries this response starts.
RoutingEntryList	RoutingInfo_t []	Array. number of elements <= RoutingTableEntries - StartIndex

Where RoutingInfo_t is defined as follows:

Parameter	***Type***	***Description***
DestinationAddress	Uint16	Network (Short) address if the route destination
Status	Uint8	Route status: bits 2-0 0x00 Active 0x01 Discovery Underway 0x02 Discovery Failed 0x03 Inactive
NextHop	Uint16	Network address of the next hop in the route

8.2.10 NWK_CHANGE_NWK_KEY_REQ

Command type: **No-response command** / **ZIGBEE_GENERIC_CNF**

This API is called by the gateway application to change the active network key to a new key.

Over the air, the key change happen in two steps: first, the new key is advertized, next, the devices are being told to start using it. The confirmation for this command is sent as soon as the key advertizing starts. Following the confirmation of this command, it may take up to a few seconds before all the devices in the network start using the new key. In order to make sure that the new key was successfully received and enabled in all the devices in the network, it is recommended to wait about 10 seconds after receiving the confirmation and before sending any other command.

8.2.10.1 Parameter list

Parameter	***Type***	***Description***
NewKey	Uint8[16]	OPTIONAL. The new key to be used. If omitted, a random key will be used.

8.2.11 NWK_GET_NWK_KEY_REQ

Command type: **No-response command** / NWK_GET_NWK_KEY_CNF

This API is called by the gateway application to request the current active ZigBee network key.

8.2.11.1 Parameter list

(This command has no parameters)

8.2.12 NWK_GET_NWK_KEY_CNF

Command type: **Specific confirmation**

This API called by the gateway subsystem to report to the gateway application the current active ZigBee network key.

8.2.12.1 Parameter list

Parameter	***Type***	***Description***
Status	Uint8	see **Status and Error Codes**
NewKey	Uint8[16]	The current active ZigBee network key

8.3 Device management API set

8.3.1 NWK_ZIGBEE_DEVICE_IND

Command type: **Incoming Indication**

The gateway subsystem hosts a database of entries for devices in the network, holding information of those devices (IEEE address, NWK address, Endpoint/Simple descriptor, status, etc). This API is called by the gateway subsystem when an entry is added to the database upon a new ZigBee device joining the network, entry removed from the database when a device leaves the network or when the information of an existing device in the database has been updated (like short address change).

For new devices, the gateway subsystem application automatically starts service discovery upon reception of an announcement message that a new device has joined the network, but notifies the gateway application that a new device has been successfully discovered only at the end of the service discovery procedure, hence abstracting it from the application developer.

8.3.1.1 Parameter list

Parameter	***Type***	***Description***
DeviceInfo	DeviceInfo_t	device information

Where DeviceInfo_t:

Parameter	***Type***	***Description***
NetworkAddress	Uint16	Address of the device in the network
IEEEAddress	Uint64	64-bit ID of the device
ParentIEEEAddress	Uint64	OPTIONAL. 64-bit ID of the parent device. This field is omitted when the parent address is unknown. Note that this field is intended for internal use, and may not reflect the actual address of the parent.
ManufacturerID	Uint16	ID of the manufacturer of the device
SimpleDescList	SimpleDescriptors_t vect[]	Array. The number of entries in this array reflect the number of endpoints on this device. This array contains the list of simple descriptors (one per endpoint)
DeviceStatus	Uint8	0 – device off line (non responding to service discovery)

		1 – device on-line 2 – device removed 255 – not applicable (this value is returned when the gateway application request information about the local gateway device)

Where SimpleDescriptors_t is defined as

Parameter	***Type***	***Description***
EndpointID	Uint8	Destination endpoint
ProfileID	Uint16	Profile ID
DeviceID	Uint16	Device ID
DeviceVersion	Uint16	Device version
InputClusterList	Uint16 array[]	Array that contains the list of Input clusters of this endpoint
OutputClusterList	Uint16 array[]	Array that contains the list of Output clusters of this endpoint

8.3.2 NWK_GET_DEVICE_LIST_REQ

Command type: **No-response command** / **NWK_GET_DEVICE_LIST_CNF**

This API is called by the gateway application to request the gateway subsystem to provide the list of devices from the device database.

8.3.2.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U}	**OPTIONAL**. if included will return just this one, otherwise return all

8.3.3 NWK_GET_DEVICE_LIST_CNF

Command type: **Specific confirmation**

This API is called by the gateway subsystem as a confirmation of the corresponding request, indicating the number of devices in the network, as maintained in the gateway subsystem database, and their description.

8.3.3.1 Parameter list

Parameter	***Type***	***Description***
Status	Uint8	see **Status and Error Codes**
DeviceList	DeviceInfo_t []	List of devices currently in the database, or a single device if specifically requested

8.3.4 NWK_DEVICE_LIST_MAINTENANCE_REQ

Command type: **No-response command** / **ZIGBEE_GENERIC_CNF**

This API is called by the gateway application to request the gateway subsystem, to re-discover services for a single device that is currently in the network/all devices in order to refresh/update the device list database. As a result of this request, the gateway server will start a service discovery procedure for that device and return indications to the gateway application via **NWK_ZIGBEE_DEVICE_IND**, if the information for an entry in the table is changed.

8.3.4.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U}	**OPTIONAL**. if included will process just this one, otherwise process all

8.3.5 NWK_REMOVE_DEVICE_REQ

Command type: **No-response command** / **ZIGBEE_GENERIC_CNF**

This API is called by the gateway application to remove the specific device from the network. The corresponding network message is sent out over the air by the gateway and the entry is cleared from the database.

8.3.5.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U}	address of the device to be removed

8.4 Device pairing API set

8.4.1 NWK_SET_BINDING_ENTRY_REQ

Command type: **Specific-response command** // **NWK_SET_BINDING_ENTRY_RSP_IND**

This API is called by the gateway application when it wants to instantiate a binding entry (or a 'pairing') /remove binding between a couple of remote devices. The corresponding OTA message must be sent out using APS acknowledged mode service so that the delivery to end node is guaranteed

8.4.1.1 Parameter list

Parameter	***Type***	***Description***
SourceAddress	AddressStruct_t {U,E}	address of the node where the binding entry shall be created (e.g. a switch)
ClusterID	Uint16	Cluster ID that is used to transport the data between the bound entries
DestAddress	AddressStruct_t {U,E}	address of the node that is the target of bound created (e.g. a light)
BindingMode	Uint8	0- Bind 1- Unbind

8.4.2 NWK_SET_BINDING_ENTRY_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem when it wants to report to the application the status of the binding request made through **NWK_SET_BINDING_ENTRY_REQ**. The API call is done by the gateway subsystem when the corresponding Bind_rsp/unbind_rsp OTA message is received.

8.4.2.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	address of the node where the binding entry was created

9 Gateway server API set

The gateway server API set is typically used to implement application functionality that allows transferring and receiving user data from end nodes (like for instance controlling door locks, reading sensors, getting notifications from alarm devices, controlling lights). The gateway server API set uses the TCP socket service access point to interface with the gateway server SW entity. API calls result in over the air frames that carry telegrams over the ZigBee network that transfer user application data (like sending on/off commands, alarms, level control etc.), using the ZigBee ZCL layer as framework for over-the-air application telegrams.

The API implementation in the gateway subsystem abstracts the formatting and parsing of the ZCL telegrams for common devices, as well as takes care of the ZCL management plane functionalities, therefore providing a very high-level abstracted interface for the application. The gateway server API has however been designed in a way where it still maintains cluster neutrality to handle use cases where application-specific telegrams must be formatted/parsed.

The gateway server API is separated into different functional domains, similar as to how the ZigBee cluster library is.

9.1 Groups and scenes API set

9.1.1 GW_ADD_GROUP_REQ

Command type: **Generic-response command**

This API is called by the gateway application to add a specific device to a group, or add an existing group of devices to another group, generating an Add Group OTA command.

9.1.1.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G,e}	
GroupID	Uint16	Group ID value
GroupName	String	Group name

9.1.2 GW_GET_GROUP_MEMBERSHIP_REQ

Command type: **Specific-response command** // **GW_GET_GROUP_MEMBERSHIP_RSP_IND**

This API is called by the gateway application to get the list of groups which a remote device belongs to.

9.1.2.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G,e}	Unicast or groupcast

9.1.3 GW_GET_GROUP_MEMBERSHIP_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to report the groups that the requested remote device belongs to and the remaining capacity for additional groups.

9.1.3.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	
Capacity	Uint8	Remaining capacity
GroupList	Uint16 array[]	Array containing the list of groups in the device

9.1.4 GW_REMOVE_FROM_GROUP_REQ
Command type: **Generic-response command**

This API is called by the gateway application to request the device or group of devices to remove their membership to a particular a group or all groups. When a group membership is removed, the scenes associated to that group shall be removed as well.

9.1.4.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G,e}	
GroupID	Uint16	OPTIONAL. Group ID value. If omitted, all group memberships will be removed from the destination device(s)

9.1.5 GW_STORE_SCENE_REQ

Command type: **Generic-response command**

This API is called by the gateway application to store a scene to a group. It is highly recommended that this is only used in groupcast mode to store a scene to the entire group that it applies to, rather than individual devices.

9.1.5.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G,e}	
GroupID	Uint16	Group ID value
SceneId	Uint8	Id of the scene

9.1.6 GW_REMOVE_SCENE_REQ

Command type: **Generic-response command**

This API is called by the gateway application to remove a scene from a group.

9.1.6.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G,e}	
GroupID	Uint16	Group ID value
SceneId	Uint8	Id of the scene

9.1.7 GW_RECALL_SCENE_REQ

Command type: **Generic-response command**

This API is called by the gateway application to recall a scene from a group. This command generates an OTA Recall Scene command

9.1.7.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G,e}	
GroupID	Uint16	Group ID value
SceneId	Uint8	Id of the scene

9.1.8 GW_GET_SCENE_MEMBERSHIP_REQ

Command type: **Specific-response command** // **GW_GET_SCENE_MEMBERSHIP_RSP_IND**

This API is called by the gateway application to get the list of scenes which a remote device belongs to.

9.1.8.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G,e}	
GroupID	Uint16	Group ID

9.1.9 GW_GET_SCENE_MEMBERSHIP_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to report the scenes that the requested remote device/group of devices have in their scene table associated with a certain group id.

9.1.9.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	Addredss of the responding device
Capacity	Uint8	Remaining capacity
GroupID	Uint16	Group ID
SceneList	Uint8 array[]	Array containing the list of scenes

9.2 Poll Control API set

9.2.1 GW_SLEEPY_DEVICE_PACKET_PENDING_REQ

Command type: **No-response command** / **ZIGBEE_GENERIC_CNF**

This API is called by the gateway application, to notify the gateway subsystem about a packet that needs to be sent to a sleepy end device. Following a call to this command, the packet_pending bit related to the respective device will be set. When the gateway subsystem receives a check-in command from the sleepy end device, and the respective packet_pending bit is set, the gateway subsystem will send a **GW_SLEEPY_DEVICE_CHECK_IN_IND** to the application, so the application can send any pending message to the device. (The ZigBee check-in response command will be sent automatically by the gateway subsystem to the end device, with payload depending on the packet_pending bit, before sending the **GW_SLEEPY_DEVICE_CHECK_IN_IND** command to the application).

The packet_pending bit is automatically cleared by the gw subsystem upon sending of the **GW_SLEEPY_DEVICE_CHECK_IN_IND** command.

Note:

- In order to receive check-in commands from a sleepy end device that supports the poll control cluster, the application must first bind the poll control cluster in the appropriate endpoint of that end device to endpoint #2 of the gateway, using NWK_SET_BINDING_ENTRY_REQ.
- The frequency in which the end device will send check-in commands can be controlled by writing directly to the CheckinInterval attribute of the endpoint that implements the poll control cluster on that device. Other related configuration - LongPollInterval and ShortPollInterval attributes - cannot be written to directly, but may be set by using GW_SEND_ZCL_FRAME_REQ to send the appropriate ZCL commands.

9.2.1.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U, E}	The address of the requested device

9.2.2 GW_SLEEPY_DEVICE_CHECK_IN_IND

Command type: **Incoming Indication**

This API is called by the gateway subsystem to notify the application that an end device is currently listening for incoming commands.

9.2.2.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U, E}	The address of the device in question

9.3 Generic foundation layer API set

9.3.1 GW_ATTRIBUTE_CHANGE_IND

Command type: **Incoming Indication**

This API is called by the ZigBee gateway subsystem to notify the gateway application that an attribute, registered within one of the endpoint(s) the gateway subsystem hosts, has been modified by a remote node in the ZigBee network. The gateway application is notified of this event so that it can 'bridge over' the command over to the non-ZigBee device. This is the case for instance of a ZigBee switch controlling a wi-fi plug via the gateway.

9.3.1.1 Parameter list

Parameter	***Type***	***Description***
EndpointID	Uint8	Local endpoint
ClusterID	Uint16	ID of the cluster which the attribute belongs to
AttributeID	Uint16	ID of the attribute that is requested to be written
AttributeType	Uint8	Type of the attribute
AttributeValue	Uint8 array[]	Attribute Data, sized according to AttributeType (as defined by the ZigBee spec.)

9.3.2 GW_GET_DEVICE_ATTRIBUTE_LIST_REQ

Command type: **Specific-response command** // **GW_GET_DEVICE_ATTRIBUTE_LIST_RSP_IND**

This API is called by the gateway application to retrieve the list of attributes registered by the gateway subsystem) for a given endpoint or the list of attributes of a remote device's endpoint. (Attributes that belong to the endpoint hosted in the gateway are registered at startup and configured through a configuration file.)

9.3.2.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/S,E}	

9.3.3 GW_GET_DEVICE_ATTRIBUTE_LIST_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to report the attribute list for the attributes registered with the gateway subsystem or a remote device, as requested.

9.3.3.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U/S}	
ClusterList	ClusterList_t array[]	Array that holds the list of clusters

Where ClusterList_t is a record which has this structure:

Parameter	***Type***	***Description***
ClusterID	Uint16	ID of the cluster which the attribute belongs to
AttributeList	Uint16 array[]	Array that holds the list of attribute IDs

9.3.4 GW_READ_DEVICE_ATTRIBUTE_REQ

Command type: **Specific-response command** // **GW_READ_DEVICE_ATTRIBUTE_RSP_IND**

This API is called by the gateway application when it wants to read out a single attribute or an array of attributes belonging to a specific cluster of a remote end node. The corresponding foundation layer command is sent out OTA as a result. The attribute is reported by the gateway subsystem via the **GW_READ_DEVICE_ATTRIBUTE_RSP_IND**.

This API is also used by the gateway application when it wants to read out a local attribute in the gateway subsystem. The gateway subsystem instantiates a local attribute table that the application may want to readout to implement application decisions (i.e. imagine the case of the gateway being used to 'bridge' a ZigBee device application message like on/off command to a non-ZigBee device acting upon the on/off command reception). The gateway subsystem must then 'mirror' the on/off server functionality. The gateway application may want to read the state of the attribute being mirrored, to display on the screen its value.

9.3.4.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B/S,E}	
ClusterID	Uint16	ID of the cluster which the attribute belongs to
AttributeList	Uint16 array[]	Array that holds the list of attribute IDs

9.3.5 GW_READ_DEVICE_ATTRIBUTE_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem when it wants to report the read out a generic attribute belonging to a specific cluster from a remote end node or as a response to a query for a local attribute readout request. If any of the requested attributes cannot be read or does not exist, the status will indicate failure, but anyhow the attributes that were successfully read will be included in AttributeRecordList (failed attributes will be omitted from this list).

9.3.5.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U/S,E}	NWK address of the remote device – 0 in case of local device
ClusterID	Uint16	ID of the cluster which the attribute belongs to
AttributeRecordList	AttributeRecord_t array[]	Array that holds the attribute information that was read

Where AttributeRecord_t is defined as

AttributeID	Uint16	ID of the attribute that is requested to be read out
AttributeType	Uint8	Type of the attribute requested
AttributeValue	Uint8 array[]	Array that contains Attribute Data, sized AttributeType

9.3.6 GW_WRITE_DEVICE_ATTRIBUTE_REQ

Command type: **Specific-response command** // **GW_WRITE_DEVICE_ATTRIBUTE_RSP_IND**

This API is called by the gateway application when it wants to write a single attribute or an array of attributes belonging to a specific cluster of a remote end node. The corresponding foundation layer command is sent out OTA as a result.

This API is also called by the gateway application when it wants to set the value of a local attribute in the gateway subsystem. The gateway subsystem instantiates a local attribute table that the application may want to set to implement application decisions.

This is useful, for example, in the following use case: imagine the gateway being used to 'bridge' a ZigBee device (like a switch) to a non-ZigBee device (like a wi-fi light) and imagine this non-ZigBee device, can be also be controlled by some other non-ZigBee devices (like wi-fi switches). When the wi-fi switch changes the on/off status of the wi-fi light, the gateway must reflect this change that happen outside of the ZigBee network, because it must 'mirror' the on/off state of the light, updating the value of the respective attribute hosted locally. Thus, if the ZigBee switch wants to know the state of the wi-fi light (whether on/off), it would issue a read attribute and the information would be readily available in the gateway.

9.3.6.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B/S,E}	
ClusterID	Uint16	ID of the cluster which the attribute belongs to
AttributeRecordList	AttributeRecord_t array[]	Array that holds the attribute information to write

9.3.7 GW_WRITE_DEVICE_ATTRIBUTE_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem when it wants to report the status of the writing of a generic attribute belonging to a specific cluster on a remote or local end node.

9.3.7.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U/S,E}	
ClusterID	Uint16	ID of the cluster which the attribute belongs to
AttributeWriteErrorList	AttributeWriteStatus_t array[]	Array that holds the attribute for which write failed. This array will be empty, if all attributes requested to be written were successfully written

Where AttributeWriteStatus_t is defined as:

Status	Uint8	Failure type
AttributeID	Uint16	ID of the attribute for which write failed

9.3.8 GW_SET_ATTRIBUTE_REPORTING_REQ

Command type: **Specific-response command** // **GW_SET_ATTRIBUTE_REPORTING_RSP_IND**

This API is called by the gateway application when it wants to configure reporting attributes belonging to a specific cluster of a remote end node. The corresponding foundation layer command is sent out as a result.

9.3.8.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	
ClusterID	Uint16	ID of the cluster which the attribute belongs to
AttributeReportList	AttributeReport_t array[]	Array that holds the list of requested attributes
MinReportInterval	Uint16	Time in secs that specify the minimum interval, in seconds, between issuing reports for the attribute specified
MaxReportInterval	Uint16	Time in secs that specify the maximum interval, in seconds, between issuing reports for the attribute specified
ReportableChange	Uint8[AttributeSize]/0	Minimum change to the attribute that results in the report being issued for 'analog' attributes. Omitted for 'digital' attributes

Where AttributeReport_t is defined as

AttributeID	Uint16	ID of the attribute that is requested to be read out
AttributeType	Uint8	Type of the attribute requested

9.3.9 GW_SET_ATTRIBUTE_REPORTING_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem when it wants to report to the gateway application the response of a specific configure reporting command sent to a remote end node

9.3.9.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	
ClusterID	Uint16	ID of the cluster which the attribute belongs to
AttributeReportConfigList	AttributeReportConfig_t array[]	Array that holds the list of attributes that could not be configured as requested. If all configurations were successful, this array will be empty.

Where AttributeReportConfig_t is defined as:

Status	Uint8	failure reason
AttributeID	Uint16	ID of the attribute that failed to be configured

9.3.10 GW_ATTRIBUTE_REPORTING_IND

Command type: **Incoming Indication**

This API is called by the gateway subsystem when it wants to report to the gateway application the report of a specific attribute, when the report attributes command is received from a remote end node.

9.3.10.1 Parameter list

Parameter	***Type***	***Description***
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	NWK address
ClusterID	Uint16	ID of the cluster which the attribute belongs to
AttributeRecordList	AttributeRecord_t array[]	Array that holds the reported attribute list

Where AttributeRecord_t is defined as in **9.3.5.1**

9.3.11 GW_SEND_ZCL_FRAME_REQ
Command type: **Generic-response command**

This API is called by the gateway application when it intends to send a raw ZCL frame over the air

9.3.11.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G,e}	Unicast or groupcast
EndpointIDSource	Uint8	Source End Point ID
ProfileID	Uint16	Profile ID, or 0xFFFF for *Don't Care* (currently, the Profile ID value set here has no effect. Instead, it is overridden by the ProfileID specified for the source endpoint in the endpoint configuration file.)
QualityOfService	Uint8	0- APS not acknowledged (best effort) 1- APS acknowledged (highly reliable)
SecurityOptions	Uint8	0- APS Security Disabled 1- APS Security Enabled
ClusterID	Uint16	Cluster ID
FrameType	Uint8	0- Frame valid across the profile 1- Frame specific to a cluster
ManufacturerSpecficFlag	Uint8	0- Non manufacturer specific 1- Manufacturer specific
ManufacturerCode	Uint16	**OPTIONAL**. This field indicates the manufacturer ID. Should be omitted if not manufacturer specific
ClientServerDirection	Uint8	0- Client to server 1- Server to client
DisableDefaultResponse	Uint8	0- Default response command will be returned 1- Default response command is disabled
SequenceNumber	Uint16	**OPTIONAL.** Cluster Transaction Sequence number. If omitted – a

		new sequence number will be internally assigned. Instead, by specifying it, a response to an incoming requests can share the same sequence number as the request.
CommandID	Uint8	CommandID
Payload	Uint8 array[]	Array that contains the ZCL payload. The actual size of this array determines the payload size.

If SequenceNumber was specified in the request, then SequenceNumber of the response will be the same. Otherwise, it is internally generated, as with most other commands.

9.3.12 GW_ZCL_FRAME_RECEIVE_IND

Command type: **Incoming Indication**

This API is called by the gateway subsystem when it wants to notify that an unprocessed ZCL frame has been received. It can be a response to a previously sent ZCL Frame, or an indication of a new incoming ZCL transaction.

9.3.12.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Cluster Sequence number, same number must be used for the default response
Address	AddressStruct_t {U,E}	Source address
ProfileID	Uint16	Profile ID
EndpointIDDest	Uint8	Destination endpoint
ClusterID	Uint16	Cluster ID
FrameType	Uint8	0- Frame valid across the profile 1- Frame specific to a cluster
ManufacturerSpecficFlag	Uint8	0- Non manufacturer specific 1- Manufacturer specific
ManufacturerCode	Uint16	If ManufacturerSpecficFlag is set to 0, this field is not present. Otherwise, this field indicates the manufacturer ID
ClientServerDirection	Uint8	0- Client to server 1- Server to client
DisableDefaultResponse	Uint8	0- Default response command must be returned 1- Default response command is disabled
CommandID	Uint8	CommandID
Payload	Uint8 array[]	Array that contains the ZCL payload. The actual size of this array determines the payload size.

9.4 Alarm API set

9.4.1 GW_ALARM_IND

Command type: **Incoming Indication**

This API is called by the gateway subsystem to indicate to the gateway application that an alarm, associated to a specific cluster in a remote device, has been generated. The alarm code is 'cluster dependent', which means that different clusters may have the same alarm code ID, therefore the application would have to carefully identify the 'reason' for the alarm associating it with the corresponding cluster 'alarm code' table

9.4.1.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U,E}	address of the device generating the alarm
AlarmCode	Uint8	Code for the Alarm. Please refer at the specific cluster description to interpret the AlarmCode in the cluster specific context
ClusterID	Uint16	Identifier of the cluster

9.4.2 GW_ALARM_RESET_REQ

Command type: **Generic-response command**

This API is called by the application to request a remote device to reset a specific alarm or all alarms.

9.4.2.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,e}	address of the destination device
AlarmCode	Uint8	OPTIONAL. Code for the Alarm. If not specified, all alarms of the destination device will be reset. Please refer at the specific cluster description to interpret the AlarmCode in the cluster specific context
ClusterID	Uint16	OPTIONAL. Identifier of the cluster. Shall be specified if and only if AlarmCode is specified.

9.5 IAS Zone device API set

9.5.1 DEV_ZONE_ENROLMENT_REQ_IND

Command type: **Incoming Request** / **DEV_ZONE_ENROLMENT_RSP**

This API is called by the gateway subsystem to indicate to the gateway application that an alarm enrolment has been received. The alarm enrolment request informs of the manufacturer code and Zone Type of the Safety and Security device wanting to enroll.

9.5.1.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Address	AddressStruct_t {U,E}	The address of the originating device
Manufacturer Code	Uint16	The Manufacturer Code field shall be the manufacturer code as held in the node descriptor for the device. Manufacturer Codes are allocated by the ZigBee Alliance.
Zone Type	Uint16	0x0000 – Standard CIE 0x000d – Motion sensor 0x0015 – Contact switch 0x0028 – Fire Sensor 0x002a – Water Sensor 0x002b – Gas Sensor 0x002c – Personal Emergence Device 0x002d – Vibration/Movement Sensor 0x010f – Remote Control 0x0115 – Key Fob 0x021d – Key Pad 0x0225 – Standard warning device (EN 50131 European Standards Series for Intruder Alarm Systems)

9.5.2 DEV_ZONE_ENROLMENT_RSP

Command type: **Outgoing response**

This API is called by the gateway application to send a response to an alarm enrolment request.

9.5.2.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number. Shall be copied from the incoming request.
Address	AddressStruct_t {U,E}	The address of the originating device, as specified in the respective indication
Enrollment Response Code	Uint8	0x00 – Success 0x01 - Zone type is not supported. 0x02 – Zone enrolment is not permitted at this time 0x03 – Zone enrolment full
Zone ID	Uint8	The Zone ID is a unique reference number allocated by the application at zone enrollment time. The Zone ID is used by alarm devices to reference specific zones when communicating with the GW. The Zone ID of each zone stays fixed until that zone is un-enrolled.

9.5.3 DEV_ZONE_STATUS_CHANGE_IND

Command type: **Incoming Indication**

This API is called by the gateway subsystem to indicate to the gateway application that the value of a ZoneStatus attribute has changed.

9.5.3.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U,E}	address of the device generating the alarm
ZoneStatus	Uint16	The current value of the ZoneStatus attribute
ExtendedStatus	Uint8	For future use.

9.6 IAS ACE device API set

9.6.1 DEV_ACE_ARM_REQ_IND

Command type: **Incoming Request** / **DEV_ACE_ARM_RSP**

This API is called by the gateway subsystem to indicate to the gateway application that a remote ACE device has sent an arm request.

9.6.1.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Address	AddressStruct_t {U,E}	The address of the originating device
ArmMode	Uint8	0x00 Disarm 0x01 Arm Day/Home Zones Only 0x02 Arm Night/Sleep Zones Only 0x03 Arm All Zones

9.6.2 DEV_ACE_ARM_RSP

Command type: **Outgoing response**

This API is called by the gateway application to send a response to an ACE device that sent an arm request.

9.6.2.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number. Shall be copied from the incoming request.
Address	AddressStruct_t {U,E}	The address of the originating device, as specified in the respective indication
ArmResponse	Uint8	0x00 All Zones Disarmed 0x01 Only Day/Home Zones Armed 0x02 Only Night/Sleep Zones Armed 0x03 All Zones Armed

9.6.3 DEV_ACE_BYPASS_IND

Command type: **Incoming Indication**

This API is called by the gateway subsystem to indicate to the gateway application that a remote ACE device has sent a bypass indication.

9.6.3.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U,E}	address of the device generating the alarm
ZoneIdList	Uint8 array[]	Array of Zone IDs. Zone ID is the index of the Zone in the CIE's zone table.

9.6.4 DEV_ACE_EMERGENCY_CONDITION_IND

Command type: **Incoming Indication**

This API is called by the gateway subsystem to indicate to the gateway application that a remote ACE device has sent an emergency condition indication.

9.6.4.1 *Parameter list*

Parameter	***Type***	***Description***
Address	AddressStruct_t {U,E}	address of the device generating the alarm
EmergencyConditionType	Uint8	2 – Emergency (general) 3 – Fire 4 - panic

9.6.5 DEV_ACE_GET_ZONE_ID_MAP_REQ_IND

Command type: **Incoming Request** / **DEV_ACE_GET_ZONE_ID_MAP_RSP**

This API is called by the gateway subsystem to indicate to the gateway application that a remote ACE device has sent a zone ID map request.

9.6.5.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Address	AddressStruct_t {U,E}	The address of the originating device

9.6.6 DEV_ACE_GET_ZONE_ID_MAP_RSP

Command type: **Outgoing response**

This API is called by the gateway application to send a response to a zone ID map request.

9.6.6.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number. Shall be copied from the incoming request.
Address	AddressStruct_t {U,E}	The address of the originating device, as specified in the respective indication
ZoneIdMapSection[N]	Uint16 array[16]	If bit n of Zone ID Map section N is set to 1, then Zone ID (16 x N + n) is allocated, else it is not allocated

9.6.7 DEV_ACE_GET_ZONE_INFORMATION_REQ_IND

Command type: **Incoming Request** / **DEV_ACE_GET_ZONE_INFORMATION_RSP**

This API is called by the gateway subsystem to indicate to the gateway application that a remote ACE device has sent a zone information request.

9.6.7.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Address	AddressStruct_t {U,E}	The address of the originating device
ZoneID	Uint8	The index of the requested zone

9.6.8 DEV_ACE_GET_ZONE_INFORMATION_RSP

Command type: **Outgoing response**

This API is called by the gateway application to send a response to a zone ID map request.

9.6.8.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number. Shall be copied from the incoming request
Address	AddressStruct_t {U,E}	The address of the originating device, as specified in the respective indication
ZoneID	Uint8	As indicated by the respective request
ZoneType	Uint16	The type of the requested zone, or 0xFFFF if this zone is unallocated
IEEE Address	Uint64	The IEEE address of the respective zone-device, or 0xFFF… if this zone is unallocated

9.7 Switchable/Dimmable/Identify device control set API

9.7.1 DEV_SET_IDENTIFY_MODE_REQ

Command type: **Generic-response command**

This API is called by the gateway application to start/stop the 'Identify' mode of the target device, sending the corresponding 'Identify' command over-the-air.

9.7.1.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	
IdentifyTime	Uint16	Time in seconds to be in identification mode: 0 – stop identify, > 0 continue identifying for identifyTime seconds remaining

9.7.2 DEV_SET_ONOFF_STATE_REQ

Command type: **Generic-response command**

This API is called by the gateway application when it wants to send an on/off command to a specific device in the network, a group of devices or broadcast. The gateway subsystem will send the corresponding ZCL command OTA

9.7.2.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	
State	Uint8	0 – off, 1 –on

9.7.3 DEV_SET_LEVEL_REQ

Command type: **Generic-response command**

This API is called by the gateway application when it wants to send a level control command to a specific device in the network, a group of devices or broadcast. The gateway subsystem will send the corresponding ZCL command

9.7.3.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	
TransitionTime	Uint16	Time to transition to the requested level in units of 100 msecs
LevelValue	Uint8	Requested Level

9.7.4 DEV_GET_LEVEL_REQ

Command type: **Specific-response command** // **DEV_GET_LEVEL_RSP_IND**

This API is called by the gateway application when it wants to know the level attribute of a node (local or remote). The gateway subsystem then sends a foundation layer command over-the-air and responds to this request via the **DEV_GET_LEVEL_RSP_IND** indication.

9.7.4.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	

9.7.5 DEV_GET_LEVEL_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to indicate the state of the level attribute of a node (local or remote), as a result of previous **DEV_SET_LEVEL_REQ**. The indication is sent by the gateway subsystem as a result of the corresponding foundation layer response being received to the gateway application.

9.7.5.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	address of the responding node
LevelValue	Uint8	Requested Level value

9.7.6 DEV_GET_ONOFF_STATE_REQ

Command type: **Specific-response command** // **DEV_GET_ONOFF_STATE_RSP_IND**

This API is called by the gateway application when it wants to know the on/off attribute of a node (local or remote). The gateway subsystem then sends a foundation layer command over-the-air and responds to this request via the **DEV_GET_ONOFF_STATE_RSP_IND** indication

9.7.6.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	

9.7.7 DEV_GET_ONOFF_STATE_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to indicate the state of the on/off attribute of a node (local or remote), as a result of previous **DEV_GET_ONOFF_STATE_REQ**. The indication is sent by the gateway subsystem as a result of the corresponding foundation layer response being received to the gateway application.

9.7.7.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	NWK address of the originator node
StateValue	Uint8	0 – OFF 1 –ON

9.7.8 DEV_SET_COLOR_REQ

Command type: **Generic-response command**

This API is called by the gateway application when it wants to send a color control command to a specific device in the network, a group of devices or broadcast. The gateway subsystem will send the corresponding ZCL command

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	
HueValue	Uint8	Requested Hue value
SaturationValue	Uint8	Requested saturation value

9.7.9 DEV_GET_COLOR_REQ

Command type: **Specific-response command** // **DEV_GET_COLOR_RSP_IND**

This API is called by the gateway application when it wants to know the Color (Hue and Saturation) attribute of a node (local or remote). The gateway subsystem then sends a foundation layer command over-the-air and responds to this request via the **DEV_GET_COLOR_RSP_IND** indication

9.7.9.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	

9.7.10 DEV_GET_COLOR_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to indicate the state of the color attributes of a node (local or remote), as a result of previous **DEV_GET_COLOR_REQ**. The indication is sent by the gateway subsystem as a result of the corresponding foundation layer response being received to the gateway application.

9.7.10.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	NWK address of the originator node
HueValue	Uint8	Read Hue value
SatValue	Uint8	Read Saturation value

9.8 Measurement sensing API set

9.8.1 DEV_GET_TEMP_REQ

Command type: **Specific-response command** // **DEV_GET_TEMP_RSP_IND**

This API is called by the gateway application to request a read of the temperature attribute of a remote device, sending out the corresponding foundation layer read attribute command. The gateway subsystem will indicate to the application the attribute reported through **DEV_GET_TEMP_RSP_IND**.

9.8.1.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	

9.8.2 DEV_GET_TEMP_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to indicate to the gateway application the read-out of the attribute requested to be read. The temperature is in DegC * 100, Where -273.15°C <= temperature <= 327.67 ºC, corresponding to a Measured Value in the range 0x954d to 0x7fff. The maximum resolution this format allows is 0.01 ºC.

9.8.2.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	
TemperatureValue	Uint16	Temperature

9.8.3 DEV_GET_POWER_REQ

Command type: **Specific-response command** // **DEV_GET_POWER_RSP_IND**

This API is called by the gateway application to request a read of the power readings of a remote device, sending out the corresponding foundation layer read attribute command. The gateway subsystem will indicate to the application the attribute reported through **DEV_GET_POWER_RSP_IND**

9.8.3.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	

9.8.4 DEV_GET_POWER_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to indicate to the gateway application the raw value of the instantaneous demand attribute of the measurement cluster at the destination device.

9.8.4.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	
PowerValue	Uint32	Power value

9.8.5 DEV_GET_HUMIDITY_REQ

Command type: **Specific-response command** // **DEV_GET_HUMIDITY_RSP_IND**

This API is called by the gateway application to request a read of the humidity readings of a remote device, sending out the corresponding foundation layer read attribute command. The gateway subsystem will indicate to the application the attribute reported through **DEV_GET_HUMIDITY_RSP_IND**.

9.8.5.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	

9.8.6 DEV_GET_HUMIDITY_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to indicate to the gateway application the read-out of the attribute requested to be read. The humidity Measured Value represents the relative humidity in % as follows:

Measured Value = 100 x Relative humidity, where 0% <= Relative humidity <= 100%, corresponding to a Measured Value in the range 0 to 0x2710.
The maximum resolution this format allows is 0.01%.

9.8.6.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	
HumidityValue	Uint16	Humidity value

9.9 Door Lock API set

9.9.1 DEV_SET_DOOR_LOCK_REQ

Command type: **Specific-response command** // **DEV_SET_DOOR_LOCK_RSP_IND**

This API is called by the gateway application to request a remote device to lock/unlock the door. It generates the corresponding OTA command

9.9.1.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G,e}	
LockMode	Uint8	0- Lock 1- Unlock
PinCodeValue	Uint8 []	Array of bytes representing the PIN/RFID code. The size of this array represents the actual length of the code. This array can be empty if Pin code is not in use.

9.9.2 DEV_SET_DOOR_LOCK_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to indicate to the gateway application that the door lock/unlock command has been received by the door lock.

9.9.2.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U}	
LockMode	Uint8	0 – lock command response 1 – unlock command response

9.9.3 DEV_GET_DOOR_LOCK_STATE_REQ

Command type: **Specific-response command** // **DEV_GET_ DOOR_LOCK_STATE_RSP_IND**

This API is called by the gateway application to request to report information about the state of the lock. The gateway subsystem will indicate to the application the attribute reported through **DEV_GET_ DOOR_LOCK_STATE_RSP_IND**

9.9.3.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B, e}	

9.9.4 DEV_GET_ DOOR_LOCK_STATE_RSP_IND

Command type: **Specific response**

This API is called by the gateway subsystem to indicate to the gateway application the state of door lock, as a response from the earlier **DEV_GET_DOOR_LOCK_STATE_REQ**.

9.9.4.1 Parameter list

Parameter	***Type***	***Description***
SequenceNumber	Uint16	Transaction Sequence Number
Status	Uint8	see **Status and Error Codes**
Address	AddressStruct_t {U,E}	
LockState	Uint8	0- Not fully locked 1- Locked 2- Unlocked
DoorState	Uint8	0- Open 1- Closed 2- Jammed 3- Forced Open

9.10 Thermostat API set

9.10.1 DEV_THERMOSTAT_SETPOINT_CHANGE_REQ

Command type: **Generic-response command**

This API is called by the gateway subsystem to increase or decrease the cool and/or the heat setpoint of a thermostat.

9.10.1.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	
Mode	Uint8	0 – adjust Heat setpoint 1 – adjust Cool setpoint 2 – adjust both setpoints
Amount	Int8	Amount of change, by increments of 0.1 deg. Celsius

9.11 Window Covering API set

9.11.1 DEV_WINDOW_COVERING_ACTION_REQ

Command type: **Generic-response command**

This API is called by the gateway subsystem to change the position/state of window covering.

9.11.1.1 Parameter list

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B,E}	
Action	Uint8	0x00 Up / Open 0x01 Down / Close 0x02 Stop 0x04 Go To Lift Value 0x05 Go to Lift Percentage 0x07 Go to Tilt Value 0x08 Go to Tilt Percentage
Value	Uint16	OPTIONAL. Shall be specified only when Action is one of 4,7
Percentage	Uint8	OPTIONAL. Shall be specified only when Action is one of 5,8

10 OTA server API set

10.1.1 OTA_UPDATE_IMAGE_REGISTERATION_REQ

Command type: No-response command / ZIGBEE_GENERIC_CNF

This API is called by the gateway application to register a new image that can be downloaded by remote devices, or to unregister a previously registered image, so it is not available for download anymore. It can also be used to update parameters (e.g. ExecutionDelay) of files that are already registered.

Note: The way the host filesystem retrieves the image from the backhaul network is outside the scope of this document

10.1.1.1 Parameter List

Parameter	***Type***	***Description***
ImagePath	String	The full path+filename of the image. The image should be located on the local file system.
RegisterUnregister	Uint8	0 – Unregister existing image. In this case – the following arguments shall NOT be specified. 1 – Registrer new image, or update the HoldExecution state.
ExecutionTimingType	Uint8	0 – execute (apply) immediately 1 – execute delay is specified 2 – execute time is specified 3 – hold execution 255 – no change (valid only if the file is already registered)
ExecutionDelay	Uint32	OPTIONAL. Shall be specified only when ExecutionTimingType == 1.
ExecutionTime	Uint32	OPTIONAL. Shall be specified only when ExecutionTimingType == 2.
UpdateSupportedDeviceList	Uint8	0 – do not change the supported device list (valid only if the file is already registered) 1 – Use the supported device list that is specified in the following

		parameter. In this case, if SupportedDeviceList is empty, the image is intended to any applicable device in the network
SupportedDeviceList	Uint64 []	Array of IEEE addresses of the devices that this image is intended for. For more details, see description of UpdateSupportedDeviceList
Notification	Uint8	0 – do not send notification 1 – broadcast notification 2 – unicast notification according to SupportedDeviceList.

10.1.2 OTA_UPDATE_ENABLE_REQ

Command type: **No-response command** / **OTA_UPDATE_ENABLE_CNF**

This API is called by the gateway application to enable or disable OTA download activity.

10.1.2.1 Parameter List

Parameter	***Type***	***Description***
Mode	UINT8	0 – OTA download enabled 1 – New OTA downloads disable. Active downloads are not interrupted. 2 – OTA download disable. Active downloads are immediately aborted.

10.1.3 OTA_UPDATE_ENABLE_CNF

Command type: **Specific confirmation**

This API is called by the gateway subsystem in response to **OTA_UPDATE_ENABLE_REQ**.

10.1.3.1 Parameter List

Parameter	***Type***	***Description***
State	Uint8	1 - SUCCESS

10.1.4 OTA_UPDATE_DOWNLOAD_FINISHED_IND

Command type: **Incoming Indication**

This API is called by the gateway subsystem to inform the application that a remote device finished/aborted downloading an image.

10.1.4.1 Parameter List

Parameter	***Type***	***Description***
Status	Uint8	0 - SUCCESS 1 - INVALID_IMAGE 2 - REQUIRE_MORE_IMAGE 3 - ABORT
Address	AddressStruct_t {U/G/B}	The address of the reporting decvice

10.1.5 OTA_UPDATE_APPLY_IMAGE_REQ

Command type: **No-response command** / **ZIGBEE_GENERIC_CNF**

This API is called by the gateway application in order to instruct remote devices that they should apply a previously downloaded image. This request must be used for images that has ExecutionTimingType set to "hold execution" (see **OTA_UPDATE_IMAGE_REGISTERATION_REQ**). It should only be used after the device reported **OTA_UPDATE_DOWNLOAD_FINISHED_IND** with status of "SUCCESS".

10.1.5.1 Parameter List

Parameter	***Type***	***Description***
Address	AddressStruct_t {U/G/B}	

11 Status and Error Codes

Set	Name	Value	Description
Network Manager and Gateway servers	STATUS_SUCCESS	0	The request accepted and/or processed successfully
	STATUS_FAILURE	1	The request cannot be performed / Generic error
	STATUS_BUSY	2	The request cannot be performed at the moment
	STATUS_INVALID_PARAMETER	3	One of the supplied argument is wrong
	STATUS_TIMEOUT	4	Timed out while waiting for an over-the-air response
OTA server	STATUS_SUCCESS	0	The request accepted and/or processed successfully
	STATUS_FAILURE	1	The request cannot be performed / Generic error

Appendix A - API command type diagrams

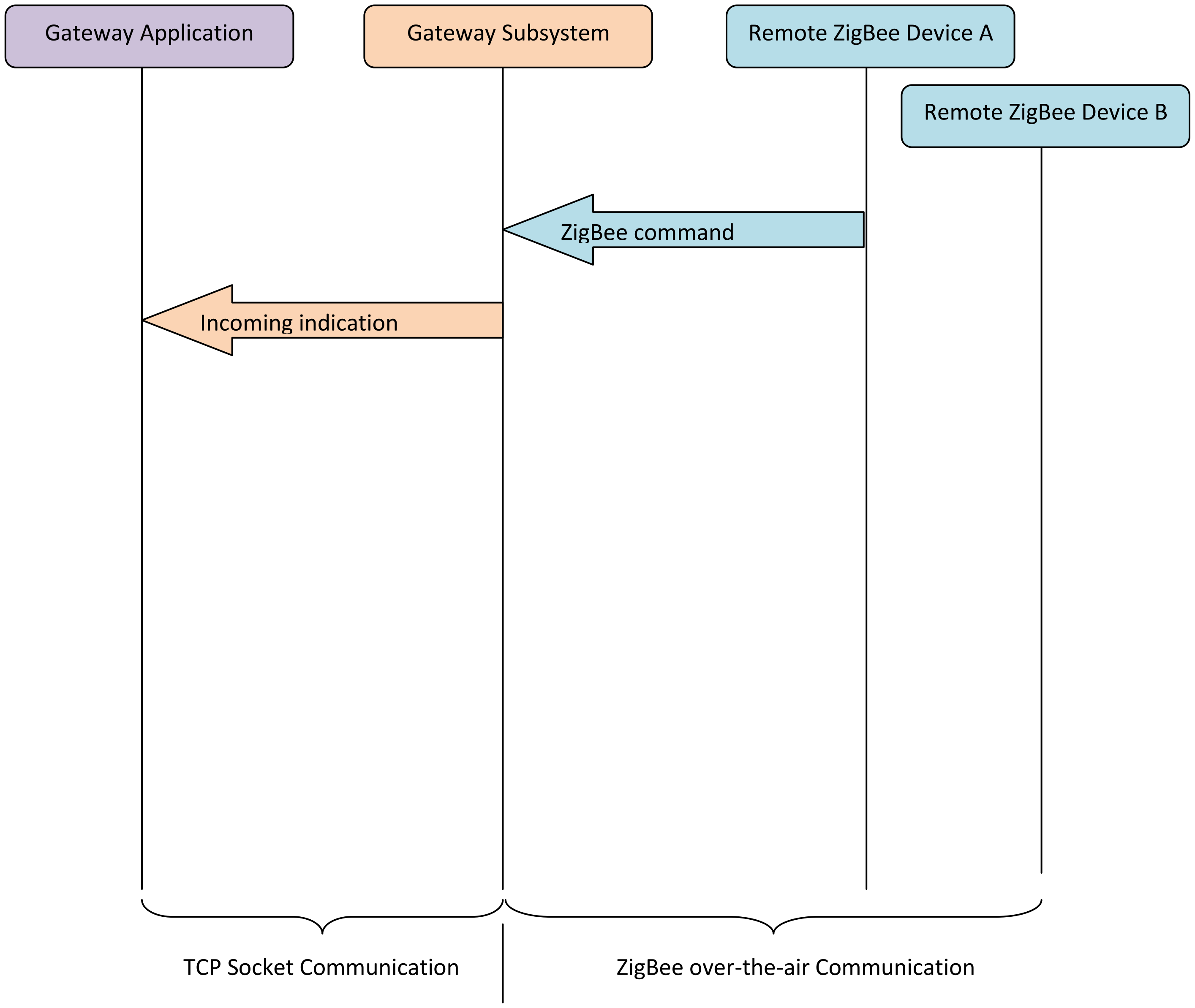

Figure 2: Incoming indication

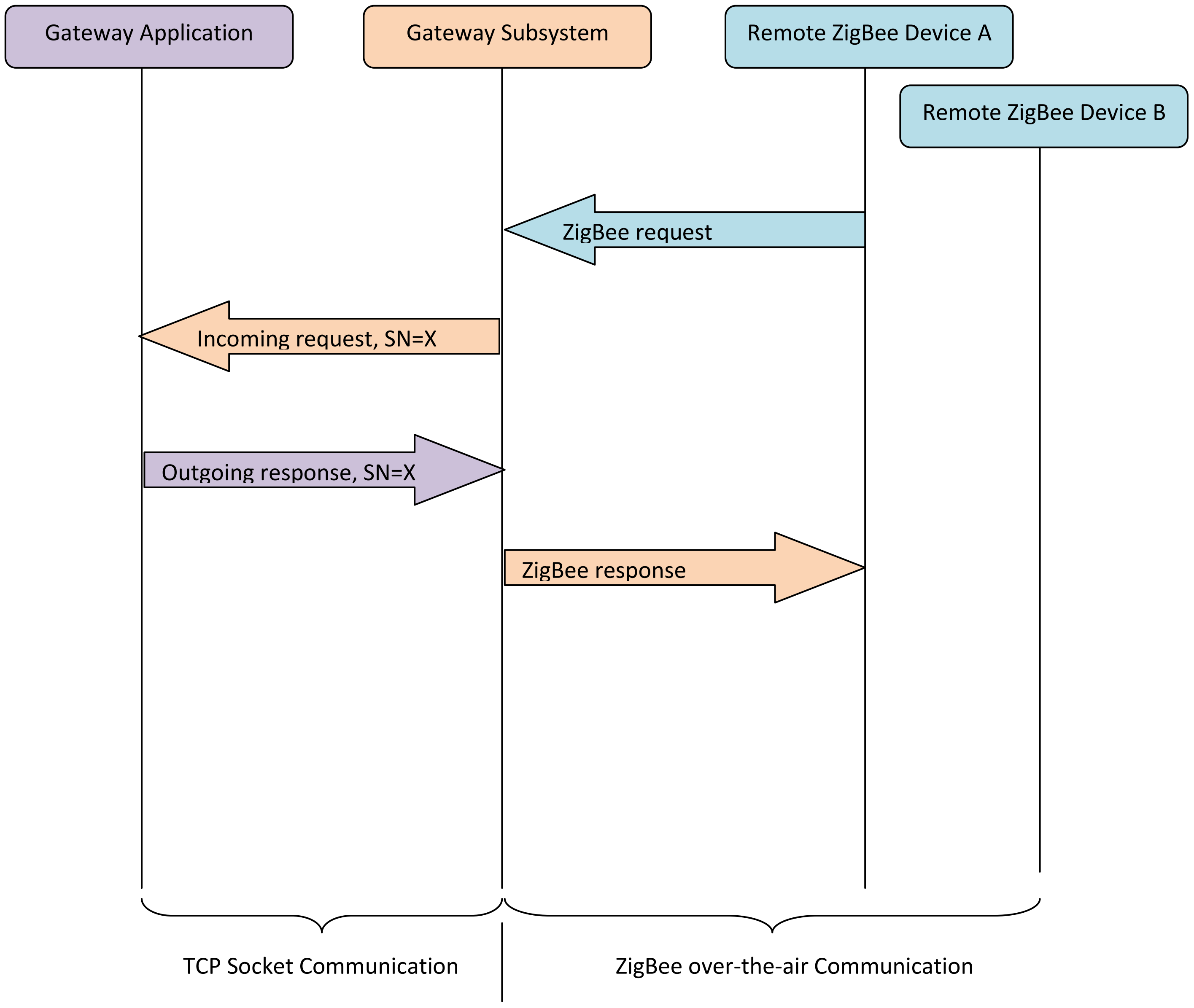

Figure 3: Incoming request + Outgoing Response

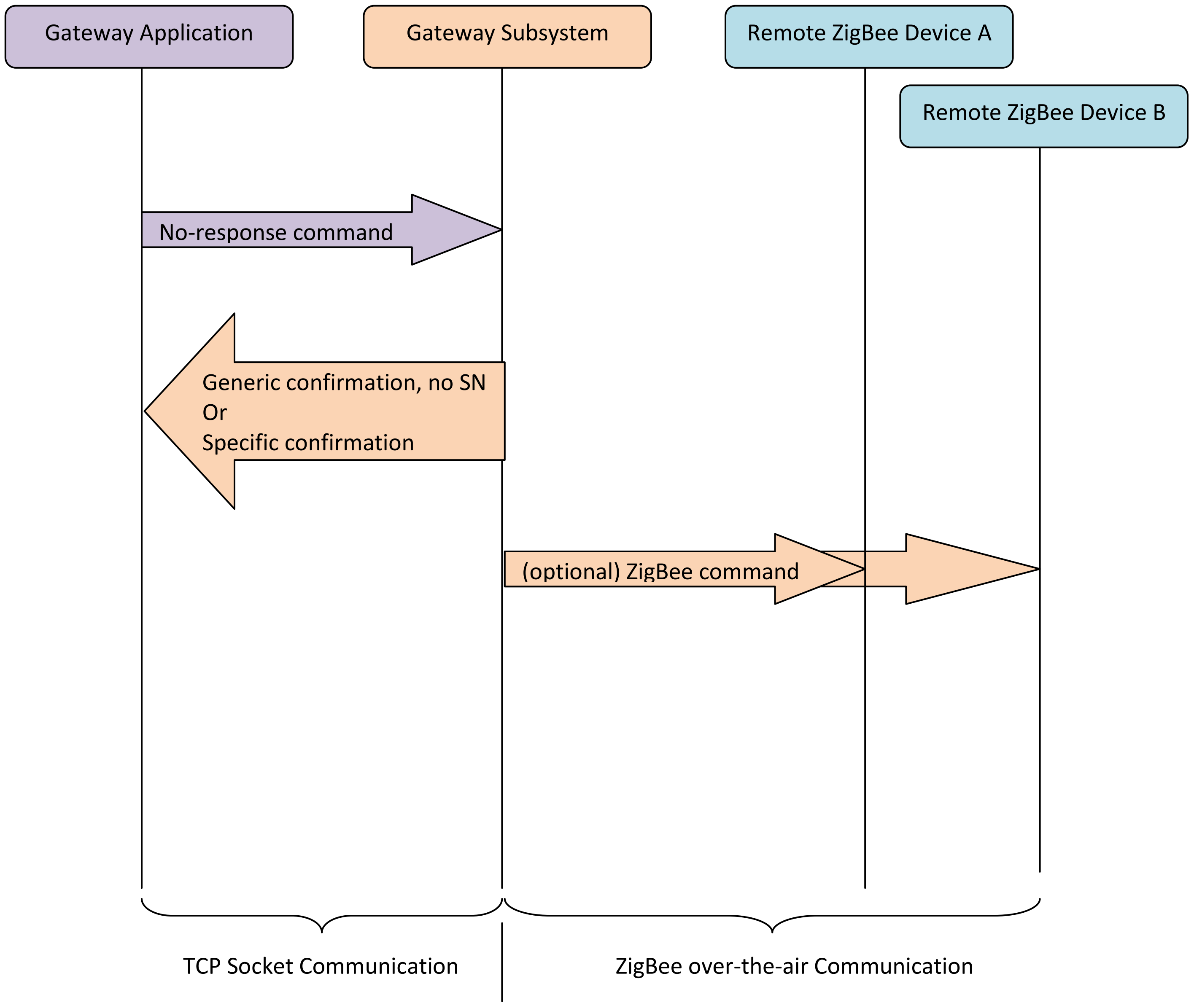

Figure 4: No-response command

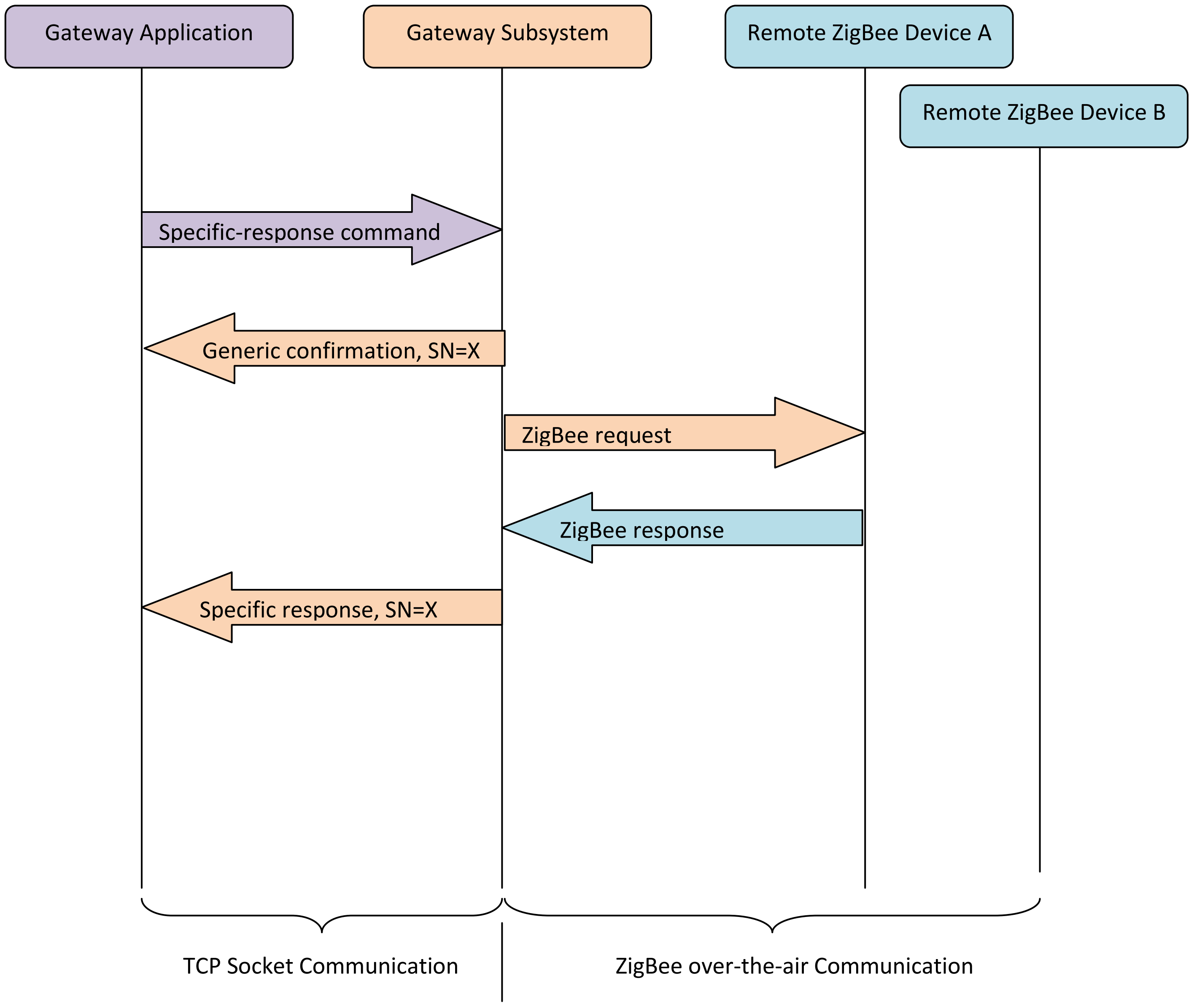

Figure 5: Specific-response command, sent as unicast

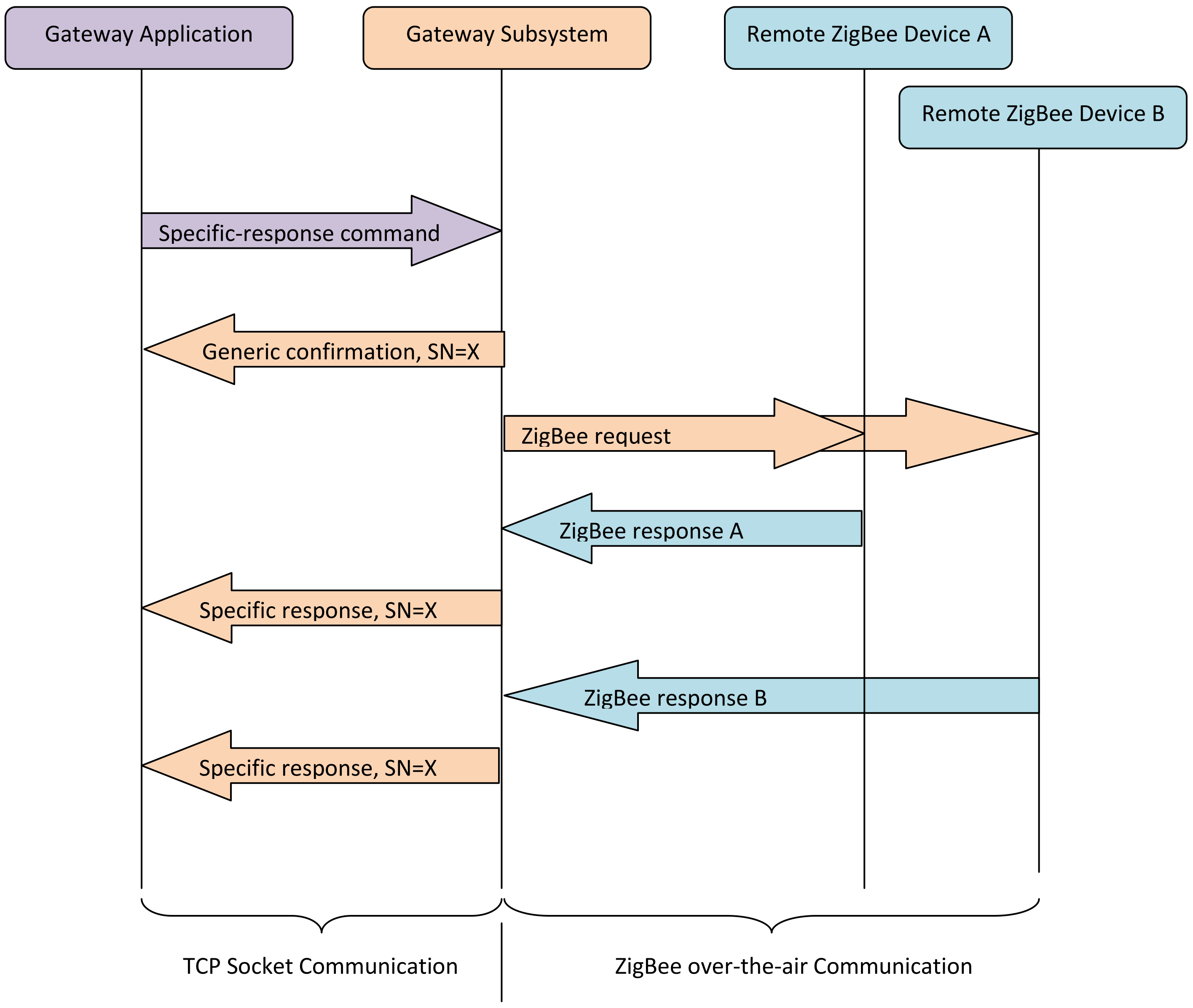

Figure 6: Specific-response command, sent as broadcast / groupcast

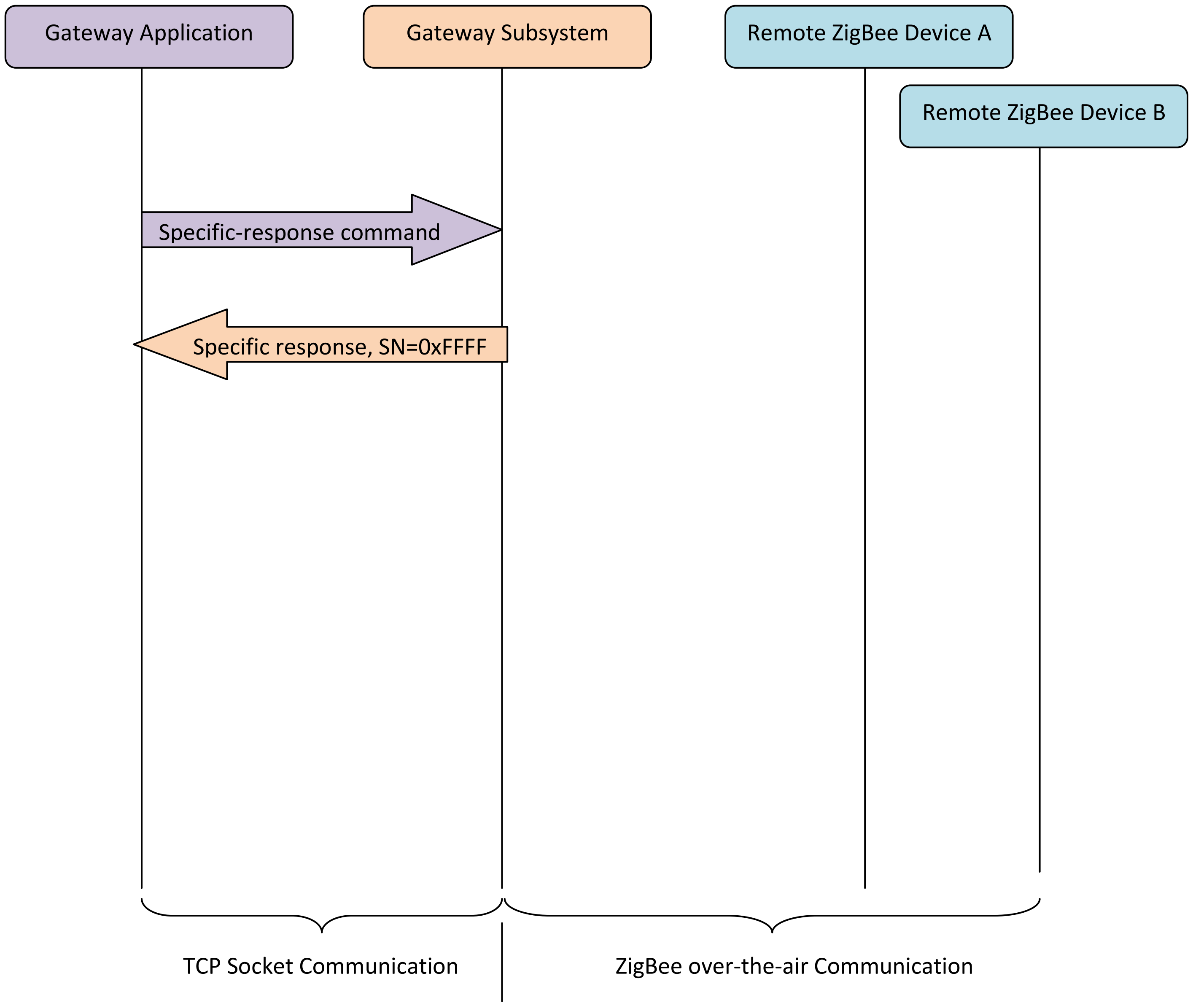

Figure 7: Specific-response command, sent to self

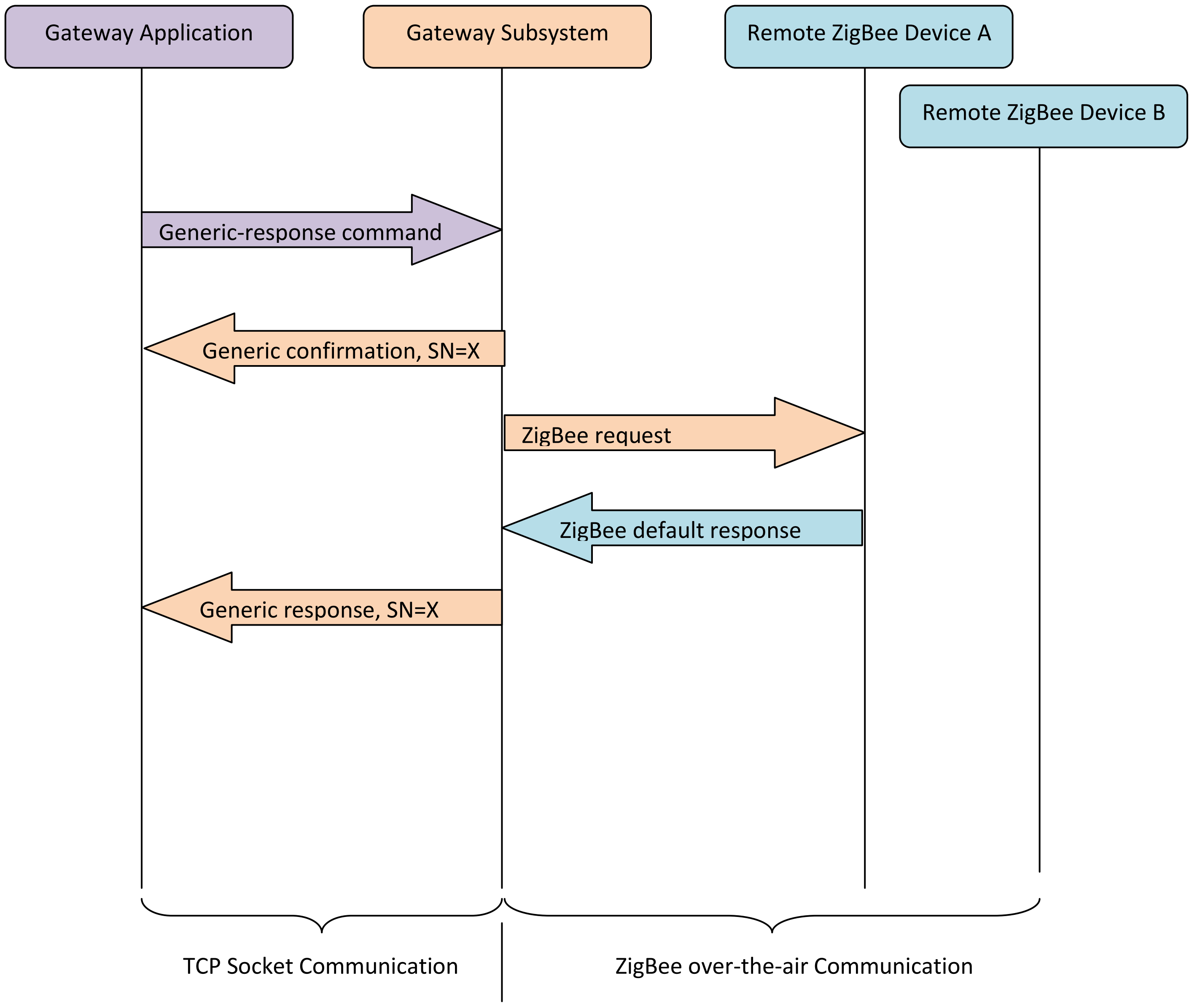

Figure 8: Generic-response command, sent as unicast

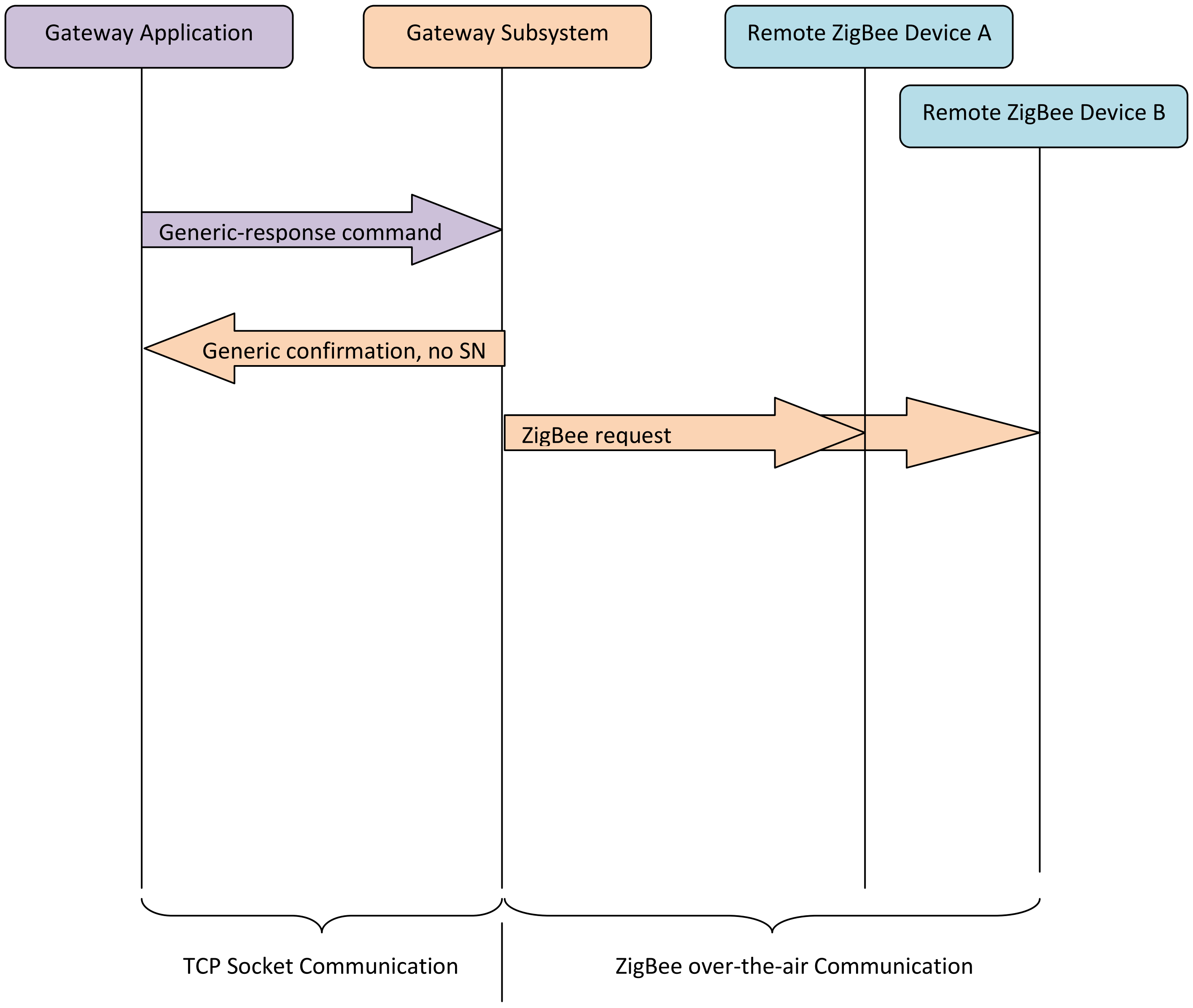

Figure 9: Generic-response command, sent as broadcast / groupcast

Appendix B – API Packet Header

For transport using a TCP socket, the protobuf-packed packets are preceded by a 4 bytes header, containing the following fields (in this order):

- **len** – 16bit number that specifies the actual length (in bytes) of the protobuf-packed packet.
- **Subsystem** – 1 byte - specify the subsystem to/from which the packet is sent/received. It can be either:
  - **18** - Network Manager Server
  - **19** - Gateway Server
  - **20** - OTA Server
- **cmd_id** – 1 byte - The command ID of the actual command being sent. This value is also available inside the packed packet. The actual command ID numbers are provided in the protobuf definition files that are part of the Z-Stack Linux Gateway package. When using command IDs in your code, always use the defined names (never hardcode the command ID numbers), as the numbers may change between releases.