RS9113 WiseConnect™

ZigBee Software Programming Reference Manual

Version 1.5.0

July 2016

Redpine Signals, Inc.
2107 N. First Street, #680
San Jose, CA 95131.
Tel: (408) 748-3385
Fax: (408) 705-2019
Email: info@redpinesignals.com
Website: www.redpinesignals.com

Disclaimer:

The information in this document pertains to information related to Redpine Signals, Inc. products. This information is provided as a service to our customers, and may be used for information purposes only.
Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating this document in the future.

About this Document

This document describes the commands to operate the RS9113-WiSeConnect Module Family for ZigBee. Host layers use RS9113-WiSeConnect module to communicate with other ZigBee devices using various profiles. Various Command requests along with the expected responses from the modules and the parameters in the commands are also described. RS9113-WiSeConnect Module Family can be operated in either End Device or Router or Co-ordinator mode at a time. This document should be referred by the developer to write software on Host to control and operate the module.

Table of Contents

Table of Figures

Table of Tables

1 ZigBee Overview

The ZigBee protocol was developed to provide low-power, wireless connectivity for a wide range of network applications concerned with monitoring and control. ZigBee is a worldwide open standard controlled by the ZigBee Alliance. ZigBee PRO is an enhancement of the original ZigBee protocol, providing a number of extra features that are particularly useful for very large networks (that may include hundreds or even thousands of nodes).

The ZigBee standard builds on the established IEEE 802.15.4 standard for packet based wireless transport. ZigBee enhances the functionality of IEEE 802.15.4 by providing flexible, extendable network topologies with integrated set-up and routing intelligence to facilitate easy installation and high resilience to failure. ZigBee networks also incorporate listen-before-talk and rigorous security measures that enable them to co-exist with other wireless technologies (such as Bluetooth and Wi-Fi) in the same operating environment.

ZigBee's wireless connectivity means that it can be installed easily and cheaply, and its built-in intelligence and flexibility allow networks to be easily adapted to changing needs by adding, removing or moving network nodes. The protocol is designed such that nodes can appear in and disappear from the network, allowing some devices to be put into a power-saving mode when not active. This means that many devices in a ZigBee network can be battery-powered, making them self-contained and, again, reducing installation costs.

The following are the basic things of ZigBee protocol.

1.1.1 ZigBee Network Nodes (Modes)

A wireless network comprises of a set of nodes that can communicate with each other by means of radio transmissions, according to a set of routing rules (for passing messages between nodes). A ZigBee wireless network includes three types of node:

1. Co-ordinator: This is the first node to be started and is responsible for forming the network by allowing other nodes to join the network through it. Once the network is established, the Co-ordinator has a routing role (is able to relay messages from one node to another) and is also able to send/receive data. Every network must have one and only one Co-ordinator.

2. Router: This is a node with a routing capability, and is also able to send/receive data. It also allows other nodes to join the network through it, so plays a role in extending the network. A network may have many Routers.

3. End Device: This is a node which is only capable of sending and receiving data (it has no routing capability). A network may have many End Devices.

1.1.2 Profiles

For the purpose of interoperability, the ZigBee Alliance has introduced the concept of a device 'profile', which contains the essential properties of a device for a particular application or market.

1.1.3 ZigBee Cluster Library (ZCL)

The ZigBee Alliance has defined the ZigBee Cluster Library (ZCL), comprising a number of standard clusters that can be applied to different functional areas. For example, all ZigBee application profiles use the Basic cluster from the ZCL.

The ZCL provides a common means for applications to communicate. It also defines attribute types (such as ints, strings, etc), common commands (e.g. for reading attributes) and default responses for indicating success or failure.

2 ZigBee Software Architecture

This section describes ZigBee software architecture and commands to operate and configure the RS9113 modules in ZigBee.

The ZigBee host mode APIs create a virtual layer of API functions, which are actually available in RS9113 ZigBee stack. The high-level architecture is shown in the following diagram.

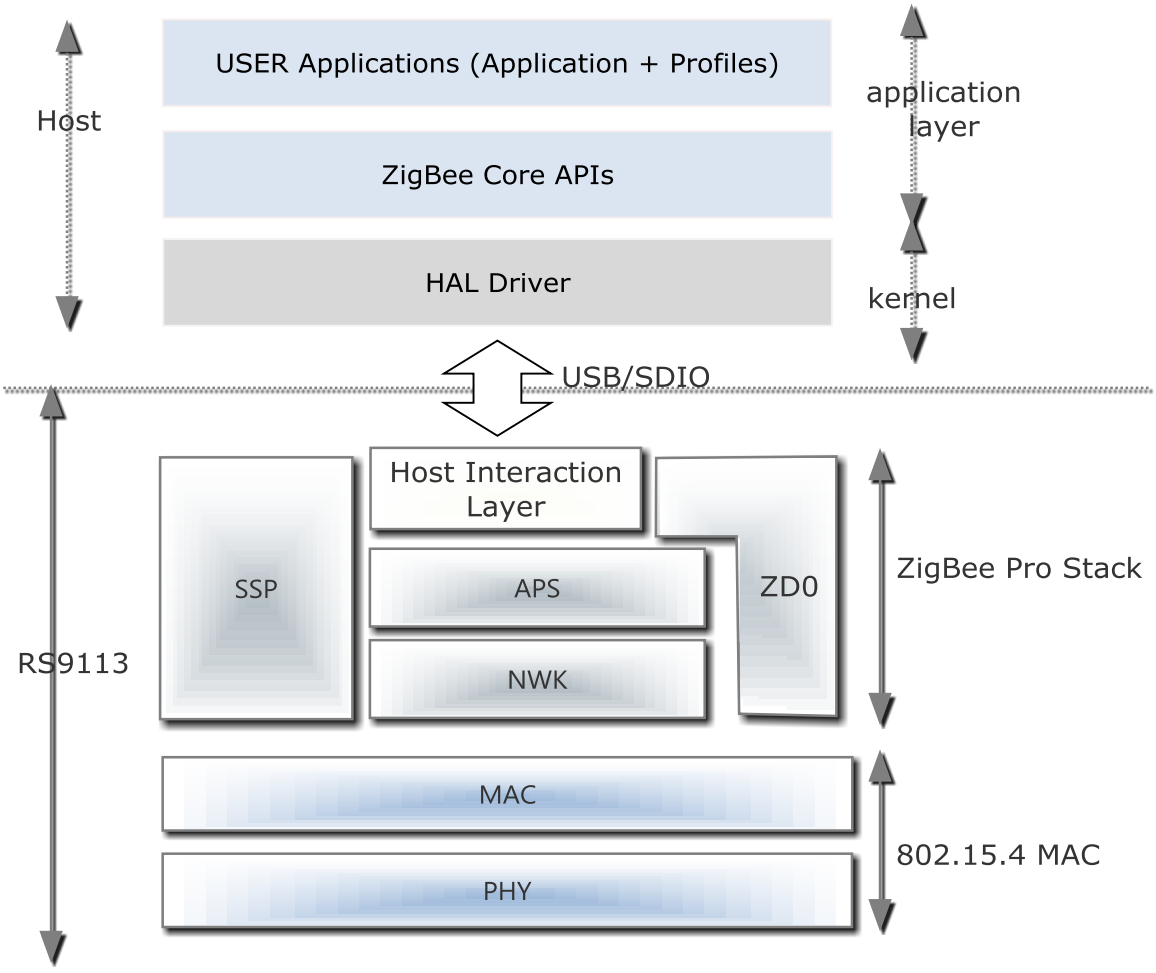

Figure 1: ZigBee Software Architecture

When the user connects the RS9113 ZigBee device to the host machine, 9113 exposes itself as USB/SDIO device. The command parser application which is running in RS9113 accepts the packets from the host and identifies the appropriate command based on the respective argument values. After identification of command and argument values, parser calls the respective API in the stack.

2.1 ZigBee Command Format

This section explains the general command format and its details. Commands should be sent to the Module in the specified format only. The commands are sent to the module and the responses are read from the module using frame write/frame read (as mentioned in the preceeding sections). These commands are called as command frames.

The format of the command frame is divided into two parts:

4. Frame descriptor
5. Frame Body(Frame body is often called as Payload)

Frame Descriptor (16 bytes)	Frame Body (multiples of 4 bytes)

Command frame format is shown below. This description is for a Little Endian System.

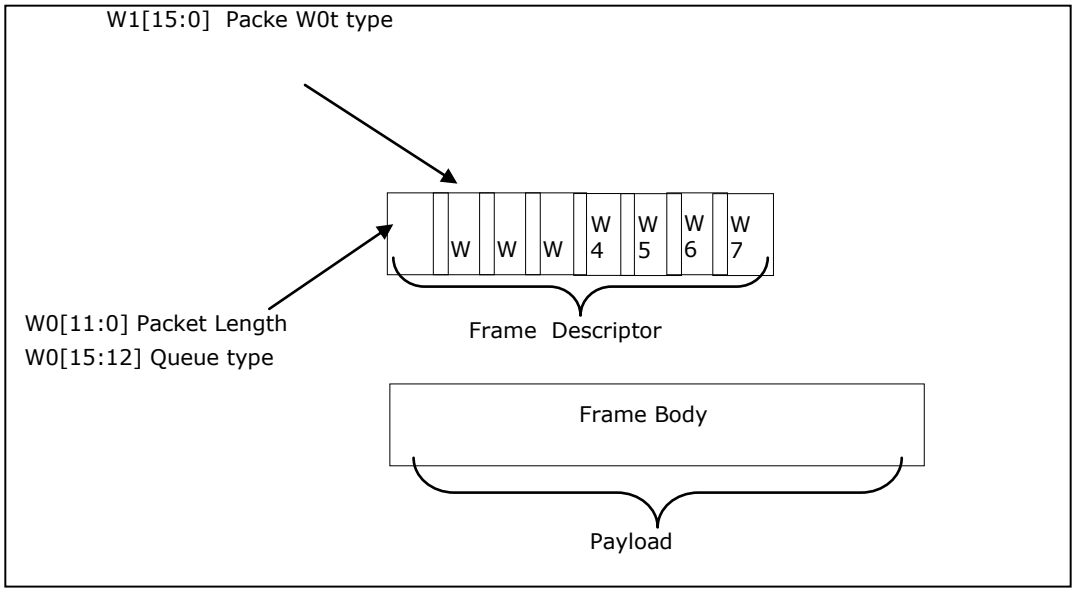

Figure 2: Command frame format

2.1.1 Frame Descriptor:

The following table provides the general description of the frame descriptor.

Word	Frame Descriptor
Word0 W0[15:0]	Bits [11:0] – Length of the frame Bits [15:12] – 1 (indicates ZigBee packet).
Word1 W1[15:0]	Reserved
Word2 W2[15:0]	Reserved
Word3 W3[15:0]	Reserved
Word4 W4[15:0]	Reserved
Word5 W5 [15:0]	Reserved
Word6 W6 [15:0]	Bits [7:0] - Reserved Bits [15:8] – Direction 1 – Host to Device 2 – Device to Host
Word7 W7 [15:0]	Bits [7:0] - Interface type Bits [15:8] - Command type

Table 1 Frame Descriptor

Three types of frames will get exchanged between the module and the host.

1. Request/Command frames - These are sent from Host to Device. Each Request/ Command has an asscociated response with it.

2. Repsonse frames – These are sent from Device to Host. These are given in response to the previous Request/Command from the Host. Each command has a single reponse.

3. Callback frames – These are sent from Module to Host. These frames are sent when

- There are multiple reponses for a particular Request/ Command frame
- There is Asynchonous message to be sent to host.

2.1.1.1 Direction

Direction is used to identify the packet directed towards to host or device. The below shown table indicates whether the packet is sent from host to device or from device to host.

Direction	Direction type
0x01	Host to Device
0x02	Device to Host

Table 2 Direction Type

2.1.1.2 Interface Type

It is used to identify the type of interface to which the frame is being redirected to in ZigBee stack.

Interface Type	Interface Id
MANAGEMENT_INTERFACE	0x01
DATA_INTERFACE	0x02
SECURITY_INTERFACE	0x03
BINDING_INTERFACE	0x04
PACKET_DATA	0x05
INTERFACE_CALLBACK(CALLBACK)	0x06

Table 3 Interface Types

2.1.1.3 Command Type

The following are the types of frame requests and responses used in ZigBee to establish communication between host and device. Command IDs for the corresponding command types are also listed in the table. The below table lists the Command, Response and Callback frames for all mode.

Interface type	Command	Cmd Id
Management	ZIGBEEFORMNETWORK	0x01
Management	ZIGBEEJOINNETWORK	0x02

Management	ZIGBEEPERMITJOIN	0x03
Management	ZIGBEELEAVENETWORK	0x04
Management	ZIGBEEFINDNETWORKANDPERFORMREJOIN	0x05
Management	ZIGBEEREJOINNETWORK	0x06
Management	ZIGBEENETWORKRESTORE	0x07
Management	ZIGBEEINITIATESCAN	0x08
Management	ZIGBEESTOPSCAN	0x09
Management	ZIGBEENETWORKSTATE	0x0A
Management	ZIGBEESTACKISUP	0x0B
Management	ZIGBEEGETSELFIEEEADDRESS	0x0C
Management	ZIGBEEISITSELFIEEEADDRESS	0x0D
Management	ZIGBEEGETSELFSHORTADDRESS	0x0E
Management	ZIGBEESETMANUFACTURERCODEFORNODEDESC	0x0F
Management	ZIGBEESETPOWERDESCRIPTOR	0x10
Management	ZIGBEESETMAXMINCOMINGTXFRSIZE	0x11
Management	ZIGBEESETMAXMOUTGOINGTXFRSIZE	0x12
Management	ZIGBEESETOPERATINGCHANNEL	0x13
Management	ZIGBEEGETDEVICETYPE	0x14
Management	ZIGBEEGETOPERATINGCHANNEL	0x15
Management	ZIGBEEGETSHORTPANID	0x16
Management	ZIGBEEGETEXTENDEDPANID	0x17
Management	ZIGBEEGETENDPOINTID	0x18
Management	ZIGBEEGETSIMPLEDDESCRIPTOR	0x19
Management	ZIGBEEGETENDPOINTCLUSTOR	0x1A
Management	ZIGBEEGETSHORTADDRFORSPECIFIEDIEEEADDR	0x1B
Management	ZIGBEESTACKPROFILE	0x1C
Management	ZIGBEEGETIEEEADDRFORSPECIFIEDSHORTADDR	0x1D
Management	ZIGBEEREADNEIGHBOURTABLEENTRY	0x1E
Management	ZIGBEEGETROUTETABLEENTRY	0x1F
Management	ZIGBEEGETREEDEPTH	0x20

Management	ZIGBEEGETNEIGHBOURTABLEENTRYCOUNT	0x21
Management	ZIGBEEGETCHILDSHORTADDRESSFORTHEINDEX	0x22
Management	ZIGBEEGETCHILDINDEXFORSPECIFIEDSHORTADDR	0x23
Management	ZIGBEEGETCHILDDETAILS	0x24
Management	ZIGBEEENDDEVICEPOLLFORDATA	0x25
Management	ZIGBEEREADCOUNTOFCHILDDEVICES	0x26
Management	ZIGBEEREADCOUNTOFROUTERCHILDDEVICE	0x27
Management	ZIGBEEGETPARENTSHORTADDRESS	0x29
Management	ZIGBEEGETPAREANTIEEEADDRESS	0x2A
Management	ZIGBEEBROADCASTNWKMANAGERREQUEST	0x2C
Management	ZDPSENDNWKADDRREQUEST	0x2D
Management	ZDPSENDIEEEADDRREQUEST	0x2E
Management	ZDPSENDDEVICEANNOUNCEMENT	0x2F
Management	ZDPSENDMATCHDESCRIPTORSREQUEST	0x30
Management	ZIGBEEACTIVEENDPOINTSREQUEST	0x31
Management	ZDPSENDPOWERDESCRIPTORREQUEST	0x32
Management	ZDPSENDNODEDESCRIPTORREQUEST	0x33
Management	ZIGBEESIMPLEDESCRIPTORREQUEST	0x34
Data	ZIGBEESENDUNICASTDATA	0x36
Data	ZIGBEESENDGROUPDATA	0x37
Data	ZIGBEEGETMAXAPSPAYLOADLENTH	0x39
Binding	ZIGBEESETBINDINGENTRY	0x3A
Binding	ZIGBEEDELETEBINDING	0x3B
Binding	ZIGBEEISBINDINGENTRYACTIVE	0x3C
Binding	ZIGBEECLEARBINDINGTABLE	0x3D
Binding	ZIGBEEBINDREQUEST	0x3E
Binding	ZIGBEEENDDEVICEBINDREQUEST	0x3F
Binding	ZIGBEEUNBINDREQUEST	0x40
Security	ZIGBEEGETKEY	0x41

Security	ZIGBEEHAVELINKKEY	0x42
Security	ZIGBEESWITCHNETWORKKEY	0x43
Security	ZIGBEEREQUESTLINKKEY	0x44
Security	ZIGBEEGETKEYTABLEENTRY	0x45
Security	ZIGBEESETKEYTABLEENTRY	0x46
Security	ZIGBEEADDORUPDATEKEYTABLEENTRY	0x47
Security	ZIGBEEFINDKEYTABLEENTRY	0x48
Security	ZIGBEEERASEKEYTABLEENTRY	0x49
Management	ZIGBEESETSIMPLEDESCRIPTOR	0x4A
Binding	ZIGBEEGETBINDINGINDICES	0x60
Management	ZIGBEEINITSTACK	0x61
Management	ZIGBEERESETSTACK	0x62
Security	ZIGBEEUPDATESAS	0x65
Security	ZIGBEEUPDATEZDO	0x66

Table 4 Command types in ZigBee

2.1.2 Event Callbacks

These interface/event specific callbacks are sent asynchronously by device to host to indicate status of Stack, Network, Data confirmation, Data Indication, Scan etc..

Interface type	Command	Cmd Id
Callback	APPSCANCOMPLETEHANDLERRESP	0x4B
Callback	APPNETWORKFOUNDHANDLERRESP	0x4D
Callback	APPZIGBEESTACKSTATUSHANDLERRESP	0x4E
Callback	APPCHILDJOINHANDLERRESP	0x4F
Callback	APPINCOMINGMANYTOONEROUTERREQUESTRESP	0x50
Callback	APPHANDLEDATAINDICATION	0x51
Callback	APPHANDLEDATACONFIRMATION	0x52

Table 5 Interface Callbacks

2.2 Operations through Host interface

This section explains the procedure that host needs to follow to send ZigBee commands frames to module and to receive responses from the module.

2.2.1 Tx Operation

Host needs to send Command frame in two parts using frame write:

1. First it is required to send 16 byte Frame descriptor
2. Optional Frame body

First frame descriptor is prepared and then frame body is appended (only if frame body is exists) at the end of frame descriptor and sent to module in a single frame write .

2.2.2 Rx Operation

The Host uses this operation:

a. To receive module's responses, for the commands issued to the module.

b. To read data received by the module from the remote terminal.

Module sends the response/received data to Host in a format as shown below:

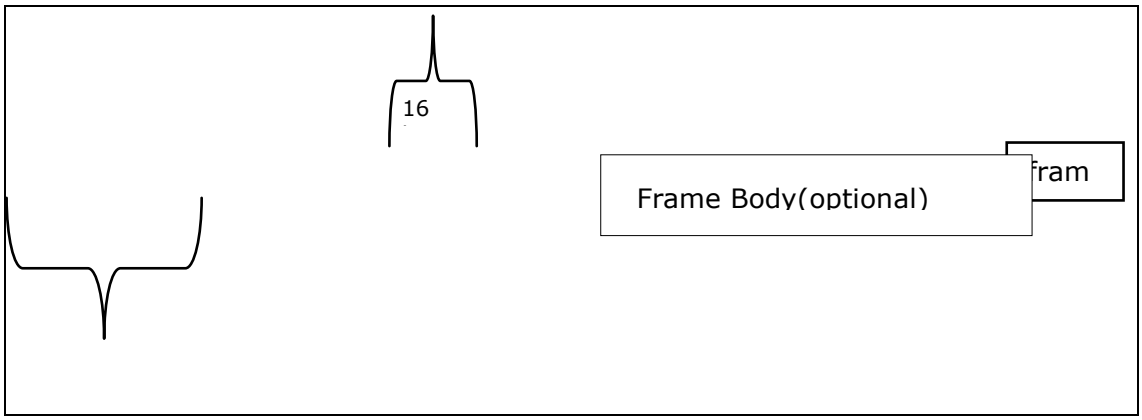

Figure 3: Rx operation descriptor and payload information

Host should follow the steps below to read the frame from the Module:

1. If any packet is pending from module, module raises an interrupt to HOST.

2. Host needs to check the reason for interrupt by reading interrupt status register using register read.
3. Read the total payload using frame read then Decode the frame descriptor and frame body.

3 ZigBee API Library

3.1 API File Organization

ZigBee APIs are organized into following directory structure

	Path(Within *RS9113.xxZ.WC.GENR.x.x.x* folder)
ZIGBEE APIs	`host/zigbee/utils/apis/core/src`
ZIGBEE Reference Applications	`host/zigbee/utils/apis/ref_apps/src`
ZIGBEE Linux Application	`host/zigbee/utils/reference_projects/src`

4 Command frames

All the command frames that are sent from host to device follow a specific format as specified in ZigBee Command Format. The 16-byte descriptor will be similar for all the tx frames and it needs to be sent for command frames. The fields that differ for each command descriptor are length of payload, Interface type and Command type. In the descriptor direction would be from host to device.

Note:

1. All the command frames expect a response frame from device to confirm that the frame is sent to device.
2. The return value for all the command frame is int16_t value. This is Success(0x1) or Failure(0x0), when a user call these APIs this return is expected.

4.1 Management frames

4.1.1 ZigBeeStackInit

Description:

When this frame is sent to device from application, then device will initialize the ZigBee Pro stack.

Supported Modes: End-device, Router and Coordinator

Prerequisites: Card ready must be received in order to issue this frame.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESTACKINIT (0x61)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only descriptor is sent but payload is not required

Payload Parameters: None

Expected Response Frames:

For this command frame ZigBeeCommandResp response frame is expected

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.2 ZigBeeDeInitStack

Description:

When this frame is sent to device from application, then device will de-init the ZigBee Pro stack and hardware initializations .

Supported Modes: End-device, Router and Coordinator

Prerequisites: none.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEDEINITSTACK (0xFF)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only descriptor is sent but payload is not required

Payload Parameters: None

Expected Response Frames:

For this command frame Card ready frame is expected.

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.3 ZigBeeStackReset

Description:

This frame allows the device to reset the ZigBee Pro stack. This frame is sent from application to device to reset all the states and reinitialize stack.

Supported Modes: End_device, Router and Coordinator

Prerequisites: The ZigBee stack must be initialized before sending this frame.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESTACKRESET (0x62)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only descriptor is sent but payload is not required

Paylaod Parameters: None

Expected Response Frames:

For this command frame ZigBeeCommandResp response frame is expected

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.4 ZigBeeUpdateSAS

Description:

This frame allows the Application to update startup attribute set required to perform internal operations in device**.**

Supported Modes: End_device, Router and Coordinator

Prerequisites: : The ZigBee stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEUPDATESAS (0x65)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t     a_extended_pan_id[8];
    uint32_t    channel_mask;
    uint8_t     startup_control;
    uint8_t     use_insecure_join;
    uint8_t     scan_attempts;
    uint8_t     parent_retry_threshold;
    uint8_t     a_trust_center_address[8];
    uint8_t     a_network_key[16];
    uint16_t    time_between_scans;
    uint16_t    rejoin_interval;
    uint16_t    max_rejoin_interval;
    uint16_t    indirect_poll_rate;
    uint16_t    a_pan_id;
    uint16_t    network_manager_address;
    uint8_t     a_trustcenter_master_key[16];
    uint8_t     a_preconfigured_link_key[16];
    uint8_t     end_device_bind_timeout;
}Startup_Attribute_Set_t
```

Paylaod Parameters:

Name	Type	Valid Range	Description
a_extended_pan_id	Integer	0x0000000000000001 – 0xfffffffffffffe	This field holds the extended PAN ID of the network. In which the device needs to be a member . If the device doesn't know the specific network, then update this 8-byte field

			with zeros otherwise specify the specific 8 bytes extended pan id.
channel_mask	Bitmap	32-bit field	The bits (b11, b12,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan) for each of the 16 valid channels.
startup_control	Integer	0x00 – 0x03	This field indicates how the device needs to respond or start depending on the startup control value. **0x00** - Indicates that the device considers itself as a part of the network. indicated by the extended PAN ID attribute. In this case device does not perform any explicit join or rejoin operation. **0x01** - Indicates that the device forms a network with extended PAN ID given by the extended PAN ID attribute. The AIB's attribute APS Designated Coordinator is set to TRUE in this case. **0x02** - Indicates that the device rejoins network with extended PAN ID given by the extended PAN ID attribute. **0x03** - Indicates that the device starts "from scratch" and join the network using association. The default value for an un-commissioned device is 0x03.
use_insecure_ join	Integer	00 = TRUE 01 = FLASE	A flag controlling the use of insecure join at startup.

scan_attempts	Integer	1 - 255	Integer value representing the number of scan attempts to make before the NWK layer decides which ZigBee coordinator or router to associate with. This attribute has default value of 5
parent_retry_ threshold	Integer	3-10	The number of failed attempts to contact a parent that will cause a "find new parent" procedure to be initiated
a_trust_center_ address	Integer	0x0000-0xFFFF	Address of the network manager.
a_network_key	Set of 16 octets	Variable	The network key.
time_between_ scans	Integer	1 – 0xFFFF	Time between scans in milliseconds
rejoin_interval	Integer	Max value:60	Rejoin interval in seconds
max_rejoin_ interval	Integer	Max value:3600	Max Rejoin interval in seconds
indirect_poll_rate	integer	In msec	The rate, in milliseconds, to poll the parent
a_pan_id	Integer	0x0000000000000001 – 0xfffffffffffffe	This field indicates the PAN ID of the device.
network_ manager_address	Integer	0x0000-0xFFFF	Address of the network manager.
a_trustcenter_ master_key	Set of 16 octets	Variable	The Trust Center master key
a_preconfigured_ link_key	Set of 16 octets	Variable	The Link key
end_device_bind _timeout	Integer	1-60	The time the coordinator will wait (in seconds) for a second end device bind

			request to arrive. The default value is 10.

Table 6 Update SAS Parameters

Expected Response Frames:

For this command frame ZigBeeCommandResp response frame is expected

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.5 ZigBeeUpdateZDO

Description:

This frame allows the Application to update the default ZDO configuration.

Supported Modes: End_device, Router and Coordinator

Prerequisites: The ZigBee stack must be initialized before calling this API.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEUPDATEZDO (0x66)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
  uint8_t     config_permit_join_duration;
  uint8_t     config_NWK_secure_all_frames;
  uint8_t     config_NWK_alt_protocol_version;
  uint8_t     config_formation_attempts;
  uint8_t     config_scan_duration;
  uint8_t     config_join_attempts;
  uint8_t     config_preconfigured_key;
  uint8_t     config_no_of_devices_joined_before_NVM_save;
  uint16_t    config_no_of_data_txns_before_NVM_save;
  uint16_t    a_config_trust_center_short_address;
  uint8_t     automatic_poll_allowed;
  uint8_t     config_authentication_poll_rate;
  uint16_t    config_switch_key_time;
  uint8_t     config_security_level;
  uint8_t     config_aps_ack_poll_time_out;
  Node_Descriptor_Information_t node_desc_info;
  uint8_t     current_powermode_avail_power_sources;
  uint8_t     current_powersource_currentpowersourcelevel;
  uint8_t     reserved[98];
}ZDO_Configuration_Table_t;
```

Paylaod Parameters:

Name	Type	Valid	Description

		Range	
config_permit_ join_duration	Integer	00-0xFF 0x00 Indicates that no devices can join 0xFF Indicates that devices are always allowed to join 0x01 - 0xFE Indicates the time in seconds for which the device allows other devices to join	defines the time for which a coordinator or router device allows other devices to join to itself.
config_NWK_ secure_all_frames	Integer	0-Enable 1-Disabled	defines whether security is applied for incoming and outgoing network data frames or not
config_NWK_ alt_protocol_ version	Integer	Default = 00	This field sets the list of protocol version numbers, other than the current protocol version number, that the device may choose to employ in a PAN that it joins. This attribute is applicable only to ZRs or ZEDs. The protocol version numbers in the list needs to refer to older versions of the ZigBee Specification.

config_formation_attempts	Integer	1	the number of times the devices attempts for formation failure.
config_scan _duration	Integer	00-0xFE	The field indicates the duration of active scan while performing startup, join or rejoin the network.
config_join_ attempts	Integer	Default = 02	This field indicates the number of times join is retried once the join fails
config_preconfigured_key	Integer	Set to 0x01 if supporting only preconfigured nwk key, or else to be set with 0x02 if we hight security value = 0x01 value = 0x02	This field indicates whether a preconfigured key is already available in the device or not
config_no_of_devices_joined_before_NVM_save	Integer	2	defines the number of devices that have to join before NVM save is performed.
config_no_of_data_txns_before_NVM_save	Integer	0x00c8	defines the number of data transmissions that has to happen before NVM save is done.
a_config_trust_center_short_add	Integer	Default 0x0000	This field holds the short address of the

ress			TC
automatic_poll_allowed	Integer	Enable-0x01 Disable-0x00(default)	This field indicates whether an end device does an auto poll or not.
config_authentication_poll_rate	Integer	Default 0x64(100 msec)	The poll rate of end device while waiting for authentication.
config_switch_key_time	Integer	Default 0x06	The time after which active key sequence number is changed, once the device receives Switch Key request
config_security_level	Integer	0x05	The security level for outgoing and incoming network frames.
config_aps_ack_poll_time_out	Integer	0xFA(250 msec)	The maximum number of seconds to wait for an acknowledgement to a transmitted frame.
node_desc_info	-		
current_powermode_avail_power_sources	Integer	8-bit	the first 4 bits of LSB gives the current sleep/ power saving mode of the node and MSB 4 bits gives the power sources available in this node

current_powerso urce_currentpow ersourcelevel	Integer	8-bit	the first 4 bit of LSB gives the current power source and 4 bits of MSB gives the current power source level.

Table 7 Update ZDO Parameters

Expected Response Frames:

For this command frame ZigBeeCommandResp response frame is expected.

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.6 ZigBeeInitiaiteScan

Description:

This frame allows the Application to initiate Scan of specified type in the provided channel mask for a specific duration. The Scan procedure is an asynchronous call.

Supported Modes: End_device, Router and Coordinator

Prerequisites: The zigBee stack must be initialized

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEINITIATESCAN (0x08)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint32_t    Channel_Mask;
    uint8_t     ScanType;
    uint8_t     Duration;
} initScanFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description

Channel Mask	Bitmap	32-bit field	The five most significant bits (b27,..., b31) are reserved. The 27 least significant bits (b0, b1,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan) for each of the 27 valid channels.
ScanType	Integer	0x00 – 0x01	00 = g_MAC_ED_SCAN_TYPE _c **01 = g_MAC_ACTIVE_SCA N_TYPE_c**
Duration	Integer	0x00-0x0e	A value used to calculate the length of time to spend scanning each channel. Thetime spent scanning each channel is (*aBaseSuperframeDuration* * (2*n* + 1)) symbols, where *n* is the value of the ScanDuration parameter .

Table 8 Initiate Scan parameters

No .of Symbols = (aBaseSuperframeDuration * ($2^{Duration}$ + 1))

Where**,**

aBaseSuperframeDuration = (**aBaseSlotDuration * aNumSuperframeSlots)**

aBaseSuperframeDuration : The number of symbols forming a superframe when the superframe order is equal to 0.

aBaseSlotDuration = 60 symbols

aNumSuperframeSlots = 16

Expected Response Frames:

For this command frame following response frames are expected:

1. Default Command response ZigBeeInitiaiteScan
2. If ScanType is **g_MAC_ACTIVE_SCAN_TYPE_c,** then following frames are expected as response frames

- Network found information event callback (AppNetworkFoundHandler) is received each and every time a new network is found. This will not be sent if there are no network during scan.

- Scan complete event callback frame(AppScanCompleteHandler) is received as response after scanning is done in each channel of the provided channel mask.

3. If ScanType is **g_MAC_ED_SCAN_TYPE_c** then energy scan complete event callback frame (AppEnergyCompleteHandler) is received as response.

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

Flow Diagram

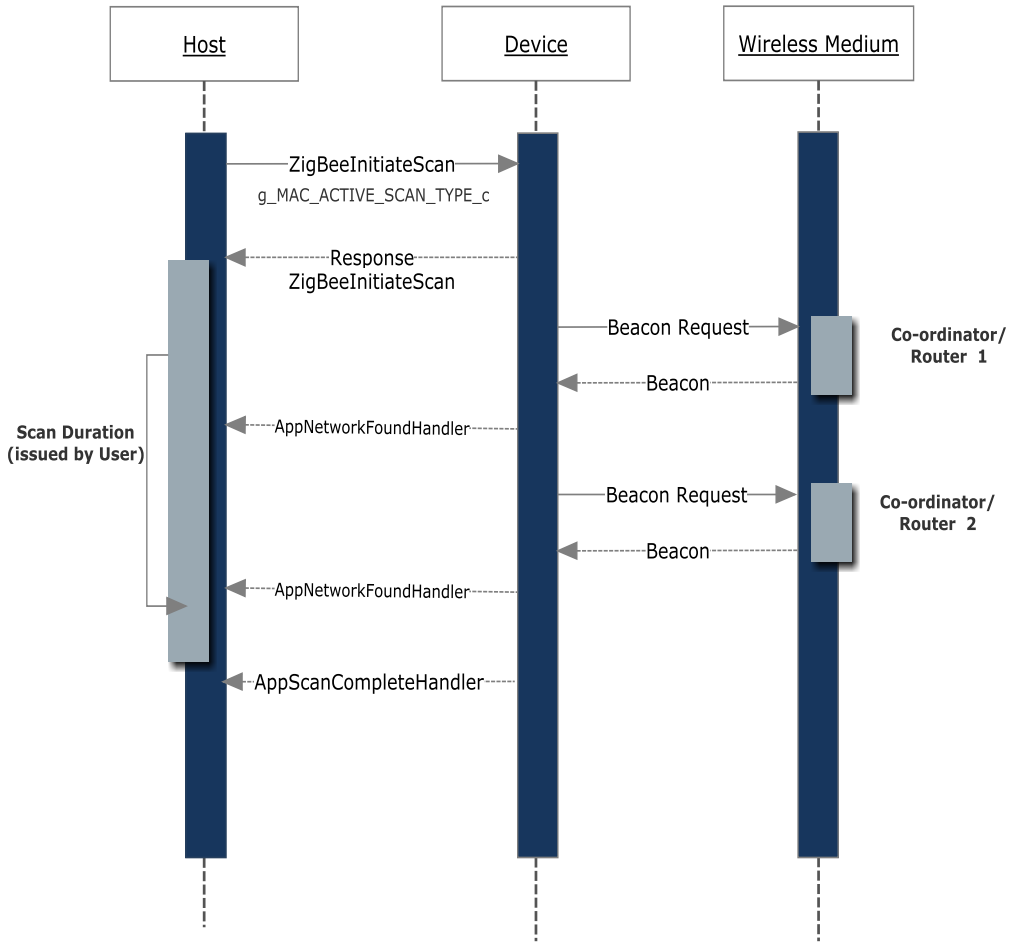

Figure 4: Scan Sequence diagram

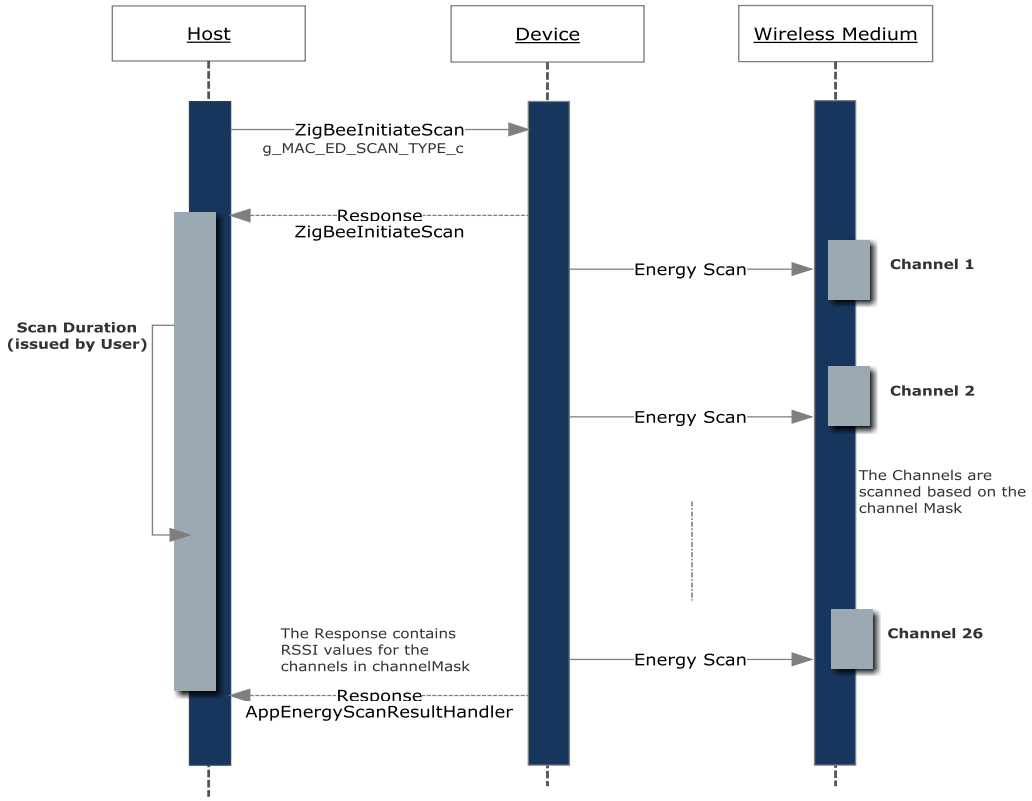

Figure 5: Energy Scan Sequence diagram

4.1.7 ZigBeeFormNetwork

Description:

This frame allows the Application to establish the Network in the provided channel with the specified Extended PAN Id. The formation procedure is an asynchronous call. The stack shall trigger AppZigBeeStackStatusHandlerResp to indicate the status of network formation to the Application.

Supported Modes: Co-ordinator.

Prerequisites: The ZigBee stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEFORMNETWORK (0x01)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t          RadioChannel;
    uint8_t          power;
    uint8_t          ExtPanId[8];
}formNetworkFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
RadioChannel	Bitmap	32-bit field	The five most significant bits (b27,..., b31) are reserved. The 27 least significant bits (b0, b1,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan) for each of the 27 valid channels.
power	Integer	0 – 18 dBm	Power setting in dBm
ExtendedPANId	Integer	0x0000000000000001 – 0xfffffffffffe	The 64-bit PAN identifier of the network to join.

Table 9 Form Network Parameters

Expected Response Frames:

For this command frame following response frames are expected:

1. For this command frame ZigBeeCommandResp response frame is expected

2. Scan complete event callback frame AppScanCompleteHandlerResp is received as response after scanning is done in each channel of the provided channel mask

3. upon establishing the network, AppZigBeeStackStatusHandlerResp shall be called by the device stack to indicate status of ZigBee Network

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.8 ZigBeeJoinNetwork

Description:

This frame allows the Application to join the Network in the provided channel with the specified Extended PAN Id. The Join procedure is an asynchronous call. The stack shall call AppZigBeeStackStatusHandlerResp to indicate the status of device joining the network to the Application

Supported Modes: End_device, Router.

Prerequisites: ZigBee stack must be initialized

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEJOINNETWORK (0x02)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t             DeviceType;
    uint8_t             RadioChannel;
    uint8_t             power;
    uint8_t             ExtPanId[8];
} joinNetworkFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Device Type	Integer	**0x00 – 0x02**	Type of device attempting to join the network. The device type can be either Router or end device 0x01 = Router 0x 02 =end device

Radio Channel	Bitmap	32-bit field	The five most significant bits (b27,..., b31) are reserved. The 27 least significant bits (b0, b1,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan) for each of the 27 valid channels.
power	Integer		Power setting in dBm
Extended PANId	Integer	0x0000000000000001 – 0xfffffffffffffffe	The 64-bit PAN identifier of the network to join.

Table 10 Join Network Parametres

Expected Response Frames:

For this command frame following response frames are expected:

1. For this command frame ZigBeeCommandResp response frame is expected
2. Upon establishing the network, AppZigBeeStackStatusHandlerResp shall be called by the stack to indicate status ZigBee Network IsUp.

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.9 ZigBeePermitJoin

Description:
This frame allows the Application to enable join permit on the device for the specified duration in seconds.

Supported Modes : Coordinator.

Prerequisites: Device must have formed network before sending this frame

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEPERMITJOIN (0x03)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t          PermitDuration;
}permitJoinFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Permit Duration	Integer	0x00 – 0xff	The length of time in seconds during which the ZigBee coordinator or router will allow associations. The value 0x00 and 0xff indicate that permission is disabled or enabled, respectively, without a specified time limit.

Table 11 permit Join Parameters

Expected Response Frames:

For this command frame ZigBeeCommandResp response frame is expected

.

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.10 ZigBeeLeaveNetwork

Description:
This frame will perform self leave from the network. The leaving procedure is an asynchronous call, but it should be part of a network to issue this frame. The stack shall trigger AppZigBeeStackStatusHandlerResp to indicate the status of device leaving the network to the Application.
Supported Modes: End_device and Router.
Prerequisites: Device must have joined a network

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEELEAVENETWORK (0x04)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only descriptor is to be sent but payload is not required

Paylaod Parameters: None

Expected Response Frames:

For this command frame ZigBeeCommandResp response frame is expected

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.11 ZigBeeFindNetworkAndPerformRejoin

Description:
The application may use this frame, 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 or rejoin. Another case is when a device has missed a Network Key update and no longer has the current Network Key.

Supported Modes: End device and Router.

Prerequisites: Device must have joined a network

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEFINDNETWORKANDPERFORMREJOIN(0x5)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint32_t            ChannelMask;
    uint8_t             Secured;
} findNWKRejoinFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Channel Mask	Bitmap	32-bit field	The five most significant bits (b27,..., b31) are reserved. The 27 least significant bits (b0, b1,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan) for each of the 27 valid channels.
Secured	Integer	0x00 – 0x01	This parameter will be TRUE (0x00) if the rejoin was performed in a secure manner. Otherwise, this parameter will be FALSE (0x01).

Table 12 Network And Perform Rejoin parameters

Expected Response Frames:

For this command frame following response frames are expected:

1. For this command frame ZigBeeCommandResp response frame is expected.
2. Upon establishing the network, AppZigBeeStackStatusHandlerResp shall be called by the stack to indicate status of ZigBee Network.

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.12 ZigBeeRejoinNetwork

Description:
The application may call this frame, 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 it wishes to a rejoin the same network.

Supported Modes: End device and Router.

Prerequisites: Device must have joined a network

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEREJOINNETWORK (0x06).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Payload Structure:

```
struct {
    uint8_t           Secured;
}rejoinNWKFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Secured	Integer	0x00 – 0x01	This parameter will be TRUE (0x00) if the rejoin was performed in a secure manner. Otherwise, this parameter will be FALSE (0x01).

Table 13 Rejoin Network parameters

Example: #define SECURED_NETWORK TRUE

Expected Response Frames:

For this command frame following response frames are expected:

1. For this command frame ZigBeeCommandResp response frame is expected
2. Upon establishing the network, AppZigBeeStackStatusHandlerResp shall be called by the stack to indicate status of ZigBee Network

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.13 ZigBeeNetworkRestore

Description:
This frame allows the Application to retrieve its network information if it was already part of the network, after retrieving network information device silently comes up in the network or does association based on the startup Control attribute in Start Up Attribute(SAS) Set.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEENETWORKRESTORE (0x07)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:

For this command frame following response frames are expected:

1. For this command frame ZigBeeCommandResp response frame is expected
2. Upon establishing the network, AppZigBeeStackStatusHandlerResp shall be called by the stack to indicate status of ZigBee Network

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.14 ZigBeeStopScan

Description:
This frame allows the Application to stop the scan that was initiated earlier.

Supported Modes: End_device, Router and co-ordinator.

Prerequisites: The device must be in scanning state.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESTOPSCAN (0x09)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:

For this command frame ZigBeeCommandResp response frame is expected

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.15 ZigBeeNetworkState

Description:
This frame allows the Application to know if the device is in the process of Joining, or already Joined or leaving the network.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEENETWORKSTATE (0x0A)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None.

Expected Response Frames:

For this command frame following response frames are expected:

1. For this command frame ZigBeeCommandResp response frame is expected

4. Upon establishing the network, AppZigBeeStackStatusHandlerResp shall be called by the stack to indicate status of ZigBee Network

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.16 ZigBeeStackIsUp

Description:
This frame is sent to know if the stack is up or not. Returns true if the stack is joined to a network and ready to send and receive messages. This reflects only the state of the local node; it does not indicate whether other nodes are able to communicate with this node.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Stack must be initialized

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESTACKISUP (0x0B)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:

For this command frame ZigBeeCommandResp response frame is expected

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.17 ZigBeeGetSelfIEEEAddress

Description:

This frame allows the Application to know the device's self extended address.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETSELFIEEEADDRESS (0x0C)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters:

Only descriptor is sent payload is not required

Expected Response Frames:

For this command frame following response frame is expected:

1. Default Command response is ZigBeeGetSelfIEEEAddressResp

API : For more information about frame usage refer corresponding API In sample application. Refer Appendix for Command type and API Name.

4.1.18 ZigBeeIsItSelfIEEEAddress

Description:

This frame allows the Application to compare the specified IEEE address with the self IEEE address. This function is implemented by stack and is invoked by Application.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEISITSELFIEEEADDRESS (0x0D)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t                ieee_Addr[8];
}isitSelfIEEEFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Ieee_Addr 0r DeviceAddress	64-bit IEEE address(Integer)	Any 64-bit IEEE address	The 64-bit IEEE address of an entity that has been added to the network.

Table 14 Self IEEE Address Parameters

Expected Response Frames:

For this command frame ZigBeeCommandResp response frame is expected

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.19 ZigBeeGetSelfShortAddress

Description:
This frame is sent to device to get the 16-bit short address of our device.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETSELFSHORTADDRESS (0x0E)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:

For this command frame following response frame is expected:

1. Default Command response ZigBeeGetSelfShortAddressResp

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.20 ZigBeeSetManufacturerCodeForNodeDesc

Description:
This frame allows the Application to specify the manufacturer code to be set in the Node descriptor.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESETMANUFACTURERCODEFORNODEDESC(0xF)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t          ManufacturerCode;
}setManufFrameSnd;
```

Paylaod Parameters:

1. **ManufacturerCode:** It indicates the 16-bit manufacturer code for the local node.

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.21 ZigBeeSetPowerDescriptor

Descriptor:

This frame allows the Application to specify the power descriptor for the device.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Stack must be initialized

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESETPOWERDESCRIPTOR (0x10).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
   setPowerDescFrameSnd  power_Desc;
} nodePowerDesc;


typsdef struct {
    uint8_t            PowerSources;
    uint8_t            CurPowerLevel;
}setPowerDescFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Power Sources	Integer	8-bit	the first 4 bits of LSB gives the current sleep/ power saving mode of the node and MSB 4 bits gives the power sources available in this node
CurPower Level	Integer	8-bit	the first 4 bit of LSB gives the current power source and 4 bits of MSB gives the current power source level.

Table 15 Power Descriptor Parameters.

Current Power Mode Value (b3b2b1b0)	Description
0000	Receiver synchronized with the receiver on when idle subfield

	of the node descriptor.
0001	Receiver comes on periodically as defined by the node power descriptor.
0010	Receiver comes on when stimulated, e.g. by a user pressing a button.
0011-1111	Reserved.

Table 16 Current Power Mode Parameters

Current Power Source Level Field (b3b2b1b0)	Charge Level
0000	Critical
0100	33%
1000	66%
1100	100%
All other values	Reserved

Table 17 Current Power Level Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.22 ZigBeeSetMaxmIncomingTxfrSize

Descriptor:

This frame allows the Application to specify the maximum incoming transfer size for the local node in uint16_t format.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESETMAXMINCOMINGTXFRSIZE (0x11).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t        MaxIncomingTxfrSize;
}setMaxIncomingTxfrFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
MaxIncomingTxfrSize	Integer	0x0000-0x7fff	This field specifies the maximum size, in octets, of the application sub-layer data unit (ASDU) that can be transferred to this node in one single message transfer.

Table 18 Incoming TXFR Size parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.23 ZigBeeSetMaxmOutgoingTxfrSize

Descriptor:

This frame allows the Application to specify the maximum outgoing transfer size for the local node.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Stack must be initialized

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESETMAXMOUTGOINGTXFRSIZE (0x12).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t        MaxOutgoingTxfrSize;
}setMaxOutTxfrFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
MaxOutgoing TxfrSize	Integer	0x0000-0x7fff	This field specifies the maximum size, inoctets, of the application sub-layer data unit (ASDU) that can be transferred from this node in one single message transfer.

Table 19 Outgoing TXFR Size Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.24 ZigBeeSetOperatingChannel

Descriptor:

This frame allows the Application to set the current channel.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Stack must be initialized

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESETOPERATINGCHANNEL (0x13).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t             Channel;
} setOperChanFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Channel	Bitmap	32-bit field	The five most significant bits (b27,..., b31) are reserved. The 27 least significant bits (b0, b1,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan) for each of the 27 valid channels.

Table 20 Operating Channel parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.25 ZigBeeGetDeviceType

Descriptor:

This frame allows the Application to get the current device type.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETDEVICETYPE (0x14).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None.

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.26 ZigBeeGetOperatingChannel

Descriptor:

When this frame is sent to device, then device returns the current operating channel.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETOPERATINGCHANNEL (0x15).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None.

Expected Response Frames:

For this command frame channel number is expected as response and that response frame is ZigBeeGetOperatingChannelResp.

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.27 ZigBeeGetShortPANId

Descriptor:

When this frame is send to device, then device returns the short PAN Id of the operating Network

Supported Modes: End_device, Router and Coordinator.

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETSHORTPANID (0x16).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None.

Expected Response Frames:

For this command frame following response frame is expected:

1. Default Command response ZigBeeGetShortPANIdResp

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.28 ZigBeeGetExtendedPanId

Descriptor:

This frame allows the Application to get the 64-bit Extended PAN id of the operating Network from device.

Supported Modes: End-device, Router and Coordinator.

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETEXTENDEDPANID (0x17).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:

For this command frame following response frame is expected:

1. Default Command response ZigBeeGetExtendedPanIdResp

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.29 ZigBeeGetEndpointId

Descriptor:

When this frame is sent to device from application, then device will send the Endpoint id located in the specified index. The index value should be less than the Number of Endpoints supported on the device.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Endpoint point information should be set earlier in device.

Endpoint info must have been set using Set Simple descriptor frame (ZigBeeSetSimpleDescriptor).

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETENDPOINTID (0x18).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t              Index;
} getEndPointIdFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Index	Integer	0x00-0xff	Index within the Active Endpoint list in the response.

Table 21 Get End Point Id Parameters

Expected Response Frames:

For this command frame following response frame is expected:

1. Default Command response ZigBeeGetEndpointIdResp

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.30 ZigBeeGetSimpleDescriptor

Descriptor:

This frame allows the Application to get the Simple descriptor for the specified endpoint id from device.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: Simple descriptor must be set earlier.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETSIMPLEDESCRIPTOR (0x19).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
   uint8_t              EndPointId;
} getSimpleDescFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
EndPointId	Integer	1-254	The 8-bit endpoint id whose simple descriptor needs to be retrieved.

Table 22 Get Simple Descriptor Parameters

Expected Response Frames:

For this command frame following response frame is expected:

1. Default Command response is ZigBeeGetSimpleDescriptorResp

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.31 ZigBeeGetEndpointCluster

Descriptor:

This frame allows the Application to read the endpoint's cluster ID from the specified index. When this frame is received by device it sends the cluster ID of that endpoint.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETENDPOINTCLUSTER (0x1A).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t                 EndPointId;
    uint8_t                 ClusterType;
    uint8_t                 ClusterIndex;
} getEPClusterFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
EndPointId	Integer	1 – 254	The 8 – bit endpoint id whose cluster id needs to be retrieved.
ClusterType	Integer	0x00 – 0x01	Indicates if the incluster list should be read or out cluster list to be read. 0 = indicates incluster list 1= indicates outcluster list.
ClusterIndex	Integer	0x 00 – 0xff	Indicates the index of the list of which cluster id is to be read.

Table 23 Get End Point Cluster Parameters

Expected Response Frames:

For this command frame following response frame is expected:

1. Default Command response ZigBeeGetEndpointClusterResp

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.32 ZigBeeGetShortAddrForSpecifiedIEEEAddr

Descriptor:

This frame allows the Application to get the 16-bit short address of the device for the given 64-bit IEEE address. The device will respond with short address of the specified IEEE address.

Supported Modes: End_device,Router and End devicce .

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETSHORTADDRFORSPECIFIEDIEEEADDR(0x1B)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
   uint8_t            ieee_Addr[8];
} getShortAddrForIeeeAddrFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
IEEE Addr	Integer (IEEE Address)	A valid 64-bit IEEE Address	Pointing to IEEE address whose 16 bit short address is to be determined.

Table 24 Addr For Specified IEEE Addr Params

Expected Response Frames:

For this command frame following response frame is expected:

1. Default Command response ZigBeeGetShortAddrForSpecifiedIEEEAddrResp

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.33 ZigBeeStackProfile

Descriptor:

This frame allows the Application to retrieve the stack profile information of device. The device will respond with stack profile information

Supported Modes: End_device,Router and End devicce .

Prerequisites: Stack must be initialized

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESTACKPROFILE (0x1C).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:

For this command frame ZigBeeStackProfileResp response frame is expected

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.34 ZigBeeGetIEEEAddrForSpecifiedShortAddr

Descriptor:

This frame allows the Application to get the 64-bit IEEE address of the device for the given 16-bit Short address.

Supported Modes: End_device,Router and End devicce .

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type – ZIGBEEGETIEEEADDRFORSPECIFIEDSHORTADDR (0x1D)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t     ShortAddr;
} getIeeeAddrForShortAddrFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
ShortAddr	Integer (Device Address List)	0x0000 - 0xffff	Provide the 16-bit short address of device whose 64-bit IEEE address need to be determined.

Table 25 IEEEAddr For Specified ShortAddr Params

Expected Response Frames:

For this command frame ZigBeeGetIEEEAddrForSpecifiedShortAddrResp response frame is expected

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.35 ZigBeeReadNeighborTableEntry

Descriptor:

This frame allows the Application to get the Neighbor table entry of specified index from device.

Supported Modes: End_device, Router and Co-ordinator.

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEREADNEIGHBORTABLEENTRY (0x1E)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t             Index;
} readNeighborTableFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Index	Integer	0x00 – 0xff	Indicates index from where the routing table entry is to be retrieved.

Table 26 Read Neighbor Table Entry Parameters

Expected Response Frames:

For this command frame ZigBeeReadNeighborTableEntryResp response frame is expected

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.36 ZigBeeGetRouteTableEntry

Descriptor:

This frame allows the Application to read the Routing table entry of specified index from device.

Supported Modes: Router and Coordinator.

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETROUTETABLEENTRY (0x1F)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t              Index;
} getRouteTableFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Index	Integer	0x 00 – 0xff	Indicates index from where the routing table entry is to be retrieved.

Table 27 Get Route Table Entry Parameters

Expected Response Frames:

For this command frame ZigBeeGetRouterTableEntryResp response frame is expected

API : For more information about frame usage refer corresponding API In sample application. Refer Appendix for Command type and API name.

4.1.37 ZigBeeTreeDepth

Descriptor:

This frame allows the Application to retrieve the current tree depth where the device has joined.

Supported Modes: End-Device , Router & Coordinator .

Prerequisites: The device must be a part of network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEETREEDEPTH (0x20).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Payload Structure:

Only Descriptor is sent, payload is not required for this frame.

Paylaod Parameters: None.

Expected Response Frames:

For this command frame ZigBeeTreeDepthResp response frame is expected

API : For more information about frame usage refer corresponding API In sample application. Refer Appendix for Command type and API name.

4.1.38 ZigBeeGetNeighborTableEntryCount

Description:

This frame allows the Application to read count of active neighbor table entries from our device.

Supported Modes: End_device, Router and Coordinator.

Prerequisites: The ZigBee stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETNEIGHBORTABLEENTRYCOUNT (0x21)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t           index;
}getRouteTableFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
Index	Integer	0x00 – 0xff	Total number of Neighbor Table entries within the Remote Device.

Table 28 Get Neighbor Table Entry Count Params

Expected Response Frames:

Command response for this request frame is ZigBeeTreeDepthResp.

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.39 ZigBeeGetChildShortAddressForTheIndex

Description:

This frame allows the Application to read the 16-bit short address of the child from the specified index.

Supported Modes: Router and Coordinator.

Prerequisites: The zigBee stack must be initialized and a child must have joined the device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETCHILDSHORTADDRESSFORTHEINDEX(0x22)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
typedef struct {
    uint8_t             ChildIndex;
}getChildShortAddrFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
Child Index	Integer	0x00 – 0xff	Index of the child device.

Table 29 Get Child Short Address For The Index

Expected Response Frames:

Default Command response is ZigBeeGetChildShortAddressForTheIndexResp.

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.40 ZigBeeGetChildIndexForSpecifiedShortAddr

Description:

This frame allows the Application to get child index for the specified 16-bit child address.

Supported Modes: Router and Coordinator.

Prerequisites: The ZigBee stack must be initialized and a child must have joined the device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETCHILDINDEXFORSPECIFIEDSHORTADDR(0x23)

Interface Type - MANAGEMENT_INTERFACE (0x1).

Payload Structure:

```
struct {
    uint8_t                 ShortAddr;
} getChildIndexForShortAddrFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
ShortAddr	Integer (Device Address List)	0x0000 - 0xffff	16-bit Short address of the child device whose index is to be determined.

Table 30 Get Child Index For Specified Short Addr

Expected Response Frames:

Default Command response ZigBeeGetChildIndexForSpecifiedShortAddrResp.

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.41 ZigBeeGetChildDetails

Description:

This frame allows the Application to get the child details from the specified child index.

Supported Modes: Router and Coordinator.

Prerequisites: The zigBee stack must be initialized and a child must have joined the device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETCHILDDETAILS (0x24)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
Struct {
    uint8_t            Index,
 } getChildDetailsFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
Child Index	Integer	0x00 – 0xff	Index of the child device.

Table 31 Get Child Details Parameters

Expected Response Frames:
For this command frame ZigBeeGetChildDetailsResp response frame is expected

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.42 ZigBeeEndDevicePollForData

Description:

This frame allows the End device application to poll the parent for data.

Supported Modes: End-Device.

Prerequisites: The zigBee stack must be initialized and the device must have joined a network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEENDDEVICEPOLLFORDATA (0x25)

Interface Type - MANAGEMENT_INTERFACE (0x1).

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:
For this command frame default ZigBeeCommandResp response frame is expected for indicating status

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.43 ZigBeeReadCountOfChildDevices

Description:

This frame allows the application to read the number of child devices on the node.

Supported Modes: Router and Coordinator.

Prerequisites: The zigBee stack must be initialized and a child should have joined the device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEREADCOUNTOFCHILDDEVICES (0x26)

Interface Type - MANAGEMENT_INTERFACE (0x1).

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:
Default Command response ZigBeeReadCountOfChildDevicesResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.44 ZigBeeReadCountOfRouterChildDevices

Description:

This frame allows the application to read the number of child devices on the node.

Supported Modes: Router and Coordinaor.

Prerequisites: The zigBee stack must be initialized and a child should have joined the device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEREADCOUNTOFROUTERCHILDDEVICES (0x27)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:
Default Command response ZigBeeReadCountOfRouterChildDevicesResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.45 ZigBeeGetParentShortAddress

Description:

This frame allows the application to read the 16-bit short address of the parent.

Supported Modes: Router and End-Device.

Prerequisites: The ZigBee stack must be initialized and the device must have joined the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETPARENTSHORTADDRESS (0x29)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:
Default Command response ZigBeeGetParentShortAddressResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.46 ZigBeeGetParentIEEEAddress:

Description:

This frame allows the application to read the 64-bit ieee address of the parent.

Supported Modes: Router and End-Device.

Prerequisites: The zigBee stack must be initialized and the device must have joined the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETPARENTIEEEADDRESS (0x2A)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:
Default Command response ZigBeeGetParentIEEEAddressResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.47 ZigBeeBroadcastNWKManagerRequest

Description:

This frame allows the application to broadcasts a request to set the identity of the network manager and the active channel mask. The mask is used when scanning for the network after missing a channel update. This request may only be sent by the current Network manager.

Supported Modes: Coordinator.

Prerequisites: The zigBee stack must be initialized and the device must have formed the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEBROADCASTNWKMANAGERREQUEST (0x2C)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
Struct {
    uint32_t          ActiveChannels,
    uint8_t           ShortAddr,
 } bcastNWKManagerFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
Active Channels	Integer	32 bit field	The new active channel mask
ShortAddr	Integer	0x0000 - 0xffff	The 16-bit short address of the network manager

Table 32 Broadcast NWKManager Request Params

Expected Response Frames:
Default Command response is ZigBeeCommandResp.

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.48 ZDPSendNWKAddrRequest

Description:

This frame allows the application to send ZDP network address request to determine the 16-bit short address of the device whose IEEE address is known.

Supported Modes: End-Device, Router and Coordinator.

Prerequisites: The zigBee stack must be initialized and the device must be in the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZDPSENDNWKADDRREQUEST (0x2D)

Interface Type - MANAGEMENT_INTERFACE (0x1).

Payload Structure:

```
Struct {
    uint8_t     ieee_Addr[8],
    uint8_t     RequestType,
    uint8_t     StartIndex
 }getZDPNWKShortAddrFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
IEEE_Addr	IEEE Address(Integer)	A valid 64-bit IEEE address	The IEEE address whose short address is to be determined.
RequestType	Integer	0x00-0xff	Request type for this command: 0x00 – Single device response 0x01 – Extended response 0x02-0xFF – reserved
StartIndex	Integer	0x00-0xff	If the Request type for this command is Extended response, the StartIndex provides the starting index for the requested elements of the associated devices list

Table 33 Network Address Request Parameters

Expected Response Frames:

The following command frames are expected:

1. For this command frame default ZigBeeCommandResp response frame is expected for indicating status and

2. AppHandleDataIndicationResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

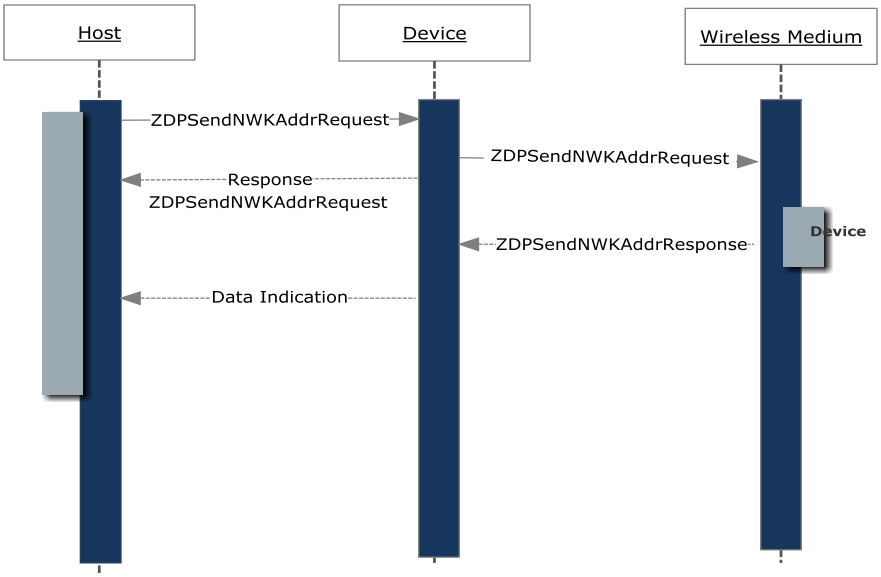

Figure 6: Network Address Request

4.1.49 ZDPSendIEEEAddrRequest

Description:

This frame allows the application to send ZDP ieee address request to determine the 64-bit IEEE address of the device whose short address is known.

Supported Modes: End-Device, Router and Coordinator.

Prerequisites: The ZigBee stack must be initialized and the device must be in the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZDPSENDIEEEADDRREQUEST (0x2E)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
Struct {
    Uint16_t    ShortAddr,
    uint8_t     RequestType,
```

```
    uint8_t     StartIndex,
    uint8_t     APSAckRequired
} getZDPIEEEAddrFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
ShortAddr	Device Address(Integer)	0x0000 - 0xffff	The short address whose IEEE address is to be determined.
RequestType	Integer	0x00-0xff	Request type for this command: 0x00 – Single device response 0x01 – Extended response 0x02-0xFF – reserved
StartIndex	Integer	0x00-0xff	If the Request type for this command is Extended response, the StartIndex provides the starting index for the requested elements of the associated devices list
APSAckRequired	Integer	0x00 – 0x01	TRUE (0x00) indicates APS ack is required.

Table 34 IEEE address Request Parameters

Expected Response Frames:
The following command frames are expected:

1. For this command frame default ZigBeeCommandResp response frame is expected for indicating status and
2. AppHandleDataIndicationResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

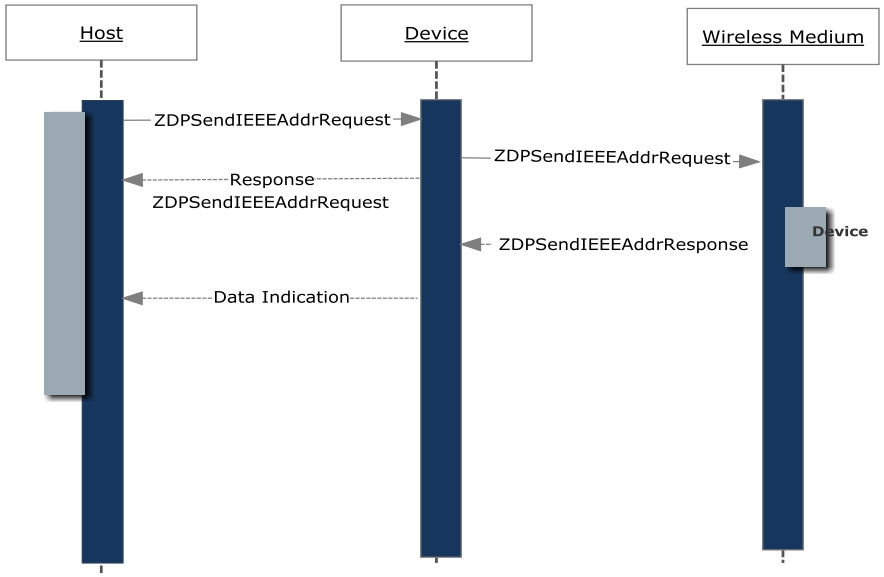

Figure 7: IEEE Address Request

4.1.50 ZDPSendDeviceAnnouncement

Description:

This frame allows the application to send a broadcast for a ZDO Device announcement. Normally, it is NOT required to send this as the stack automatically sends a device announcement during joining or rejoining, as per the spec. However, if the device wishes to broadcast device announcement it can do through this frame.

Supported Modes: End-Device, Router.

Prerequisites: The ZigBee stack must be initialized and the device must be in the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZDPSENDDEVICEANNOUNCEMENT (0x2F)

Interface Type - MANAGEMENT_INTERFACE (0x1).

Payload Structure:

Only Descriptor is sent, payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.51 ZigBeeSetSimpleDescriptor

Description:

This frame allows the application to set simple descriptor request, which contains information about profile, In and Out Clusters of a specific endpoint.

Supported Modes: End_Device, Router and Coordinator.

Prerequisites: The ZigBee stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESETSIMPLEDESCRIPTOR (0x4A)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t    ProfileId;
    uint16_t    DevId;
    uint8_t     EndPointId;
    uint8_t     DevVersion;
    uint8_t     InClusterCnt;
    uint8_t     OutClusterCnt;
    uint8_t     ClusterInfo[40];
} setSimpleDescFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
ProfileId	Integer	0x0000 – 0xffff	The Endpoint on which these clusters are defined
DevId	Integer	0x0000 - 0xfff7	The address of the designated network channel manager function
EndPointId	Integer	1-254	The endpoint on the destination.
DevVersion	Integer	0x00 – 0x0f	The version of the ZigBee protocol in use in the discovered network.
InClusterCnt	Integer	0x00-0xff	The number of Input Clusters
OutClusterCnt	Integer	0x00-0xff	The number of Output Clusters
ClusterInfo	Integer	-	Buffer for input and output cluster, supports

			20 in and 20 out clusters per endpoint

Table 35 Set Simple Descriptor Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.1.52 ZDPSendMatchDescriptorsRequest

Description:

This frame allows the application to send Match Descriptor request on air to know which other devices are supported to start data communication.

Supported Modes: End-Device, Router and Coordinator.

Prerequisites: The ZigBee stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZDPSENDMATCHDISCRIPTORREQUEST (0x30)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t    ShortAddr;
    uint16_t    ProfileId;
    uint16_t    DestAddress;
    uint8_t     InClusterCnt;
    uint8_t     OutClusterCnt;
    uint8_t     APSAckRequired;
    uint8_t     ClusterInfo[40];
}sendMatchDescFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
ShortAddr	Device Address(Integer)	0x0000 - 0xffff	The device short address whose matching end-points are desired.
ProfileId	Integer	0x0000 – 0xffff	The application profile id
DestAddress	Integer	0x0000 - 0xffff	Destination device address

InClusterCnt	Integer	0x00 – 0x0f	The number of Input Clusters
OutClusterCnt	Integer	0x00-0xff	The number of output Clusters
APSAckRequired	Integer	0x00 – 0x01	TRUE (0x00) indicates APS ack is required.
ClusterInfo	Integer	-	Buffer for input and output cluster, supports 20 in and 20 out clusters per endpoint

Table 36 Match Descriptor Request Parameters

Expected Response Frames:
The following command frames are expected:

1. For this command frame default ZigBeeCommandResp response frame is expected for indicating status and

2. AppHandleDataIndicationResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

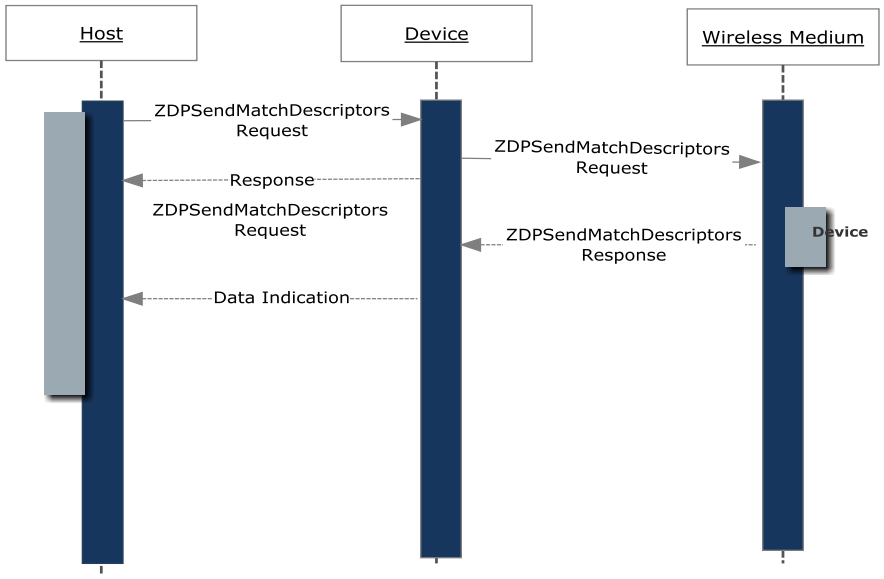

Figure 8: Match Descriptor Request

4.1.53 ZigBeeActiveEndpointsRequest

Description:

This frame allows the application to send ZDP Active Endpoint request to specified short address.

Supported Modes: End-Device, Router and Coordinator.

Prerequisites: The ZigBee stack must be initialized and the device must be in the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEACTIVEENDPOINTSREQUEST (0x31)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t    ShortAddr;
    uint8_t     APSAckRequired;
} activeEPOfShortAddrFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
ShortAddr	Integer	0x0000 - 0xffff	The device short address whose matching end-points are desired.
APSAck Required	Integer	0x00 – 0x01	TRUE (0x00) indicates APS ack is required.

Table 37 Active End Point Request Parameters

Expected Response Frames:
The following command frames are expected:

1. For this command frame default ZigBeeCommandResp response frame is expected for indicating status

2. AppHandleDataIndicationResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

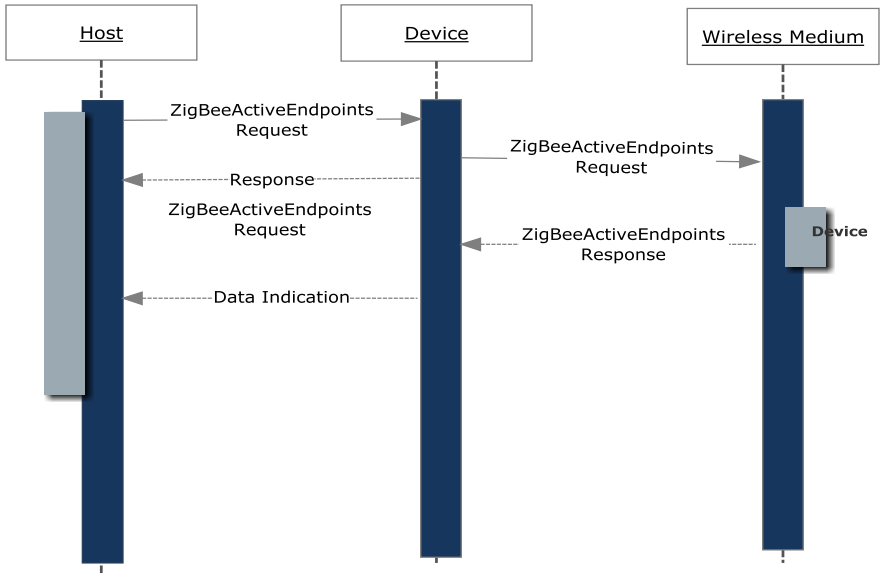

Figure 9: Active Endpoint Request

4.1.54 ZDPSendPowerDescriptorRequest

Description:

This frame allows the application to send ZDP power descriptor request to the specified short address.

Supported Modes: End-Device, Router and Coordinator.

Prerequisites: The ZigBee stack must be initialized and the device must be in the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZDPSENDPOWERDESCRIPTORREQUEST (0x32)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t    ShortAddr;
    uint8_t     APSAckRequired;
} powerDescFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
ShortAddr	Integer	0x0000 - 0xffff	The device short address whose power descriptor is to be obtained..

APS Ack Required	Integer	0x00 – 0x01	TRUE (0x00) indicates APS ack is required.

Table 38 Power Descriptor Request Parameters

Expected Response Frames:
The following command frames are expected:

1. For this command frame default ZigBeeCommandResp response frame is expected for indicating status and
2. AppHandleDataIndicationResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

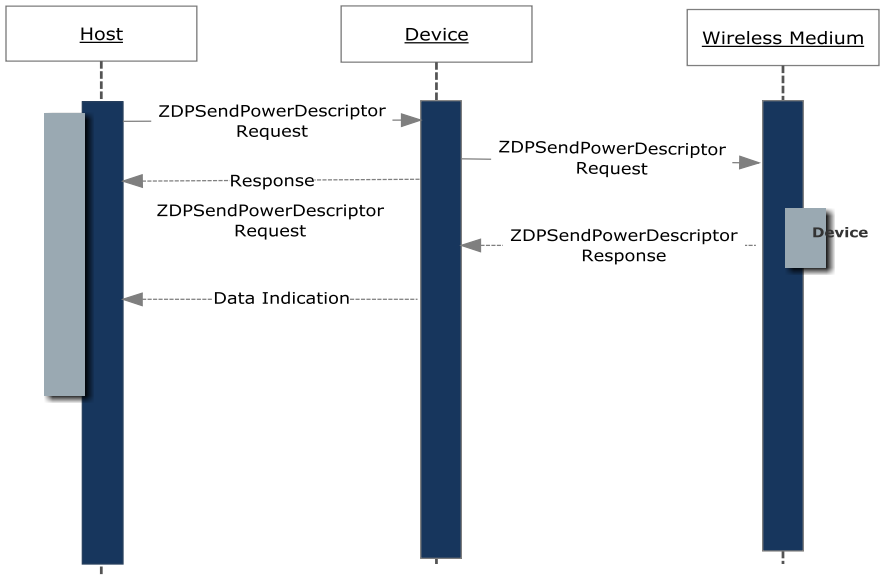

Figure 10: Power Descriptor Request

4.1.55 ZDPSendNodeDescriptorRequest

Description:

This frame allows the application to send ZDP node descriptor request.

Supported Modes: End-Device, Router and Coordinator.

Prerequisites: The zigBee stack must be initialized and the device must be in the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZDPSENDNODEDESCRIPTORREQUEST (0x33)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t          ShortAddr;
    uint8_t           APSAckRequired;
} nodeDescFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
ShortAddr	Integer	0x0000 - 0xffff	The device short address whose node descriptor is to be obtained.
APSAck Required	Integer	0x00 – 0x01	TRUE (0x00) indicates APS ack is required.

Table 39 Node Descriptor Request Parameters

Expected Response Frames:

The following command frames are expected:

1. For this command frame default ZigBeeCommandResp response frame is expected for indicating status and
2. AppHandleDataIndicationResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

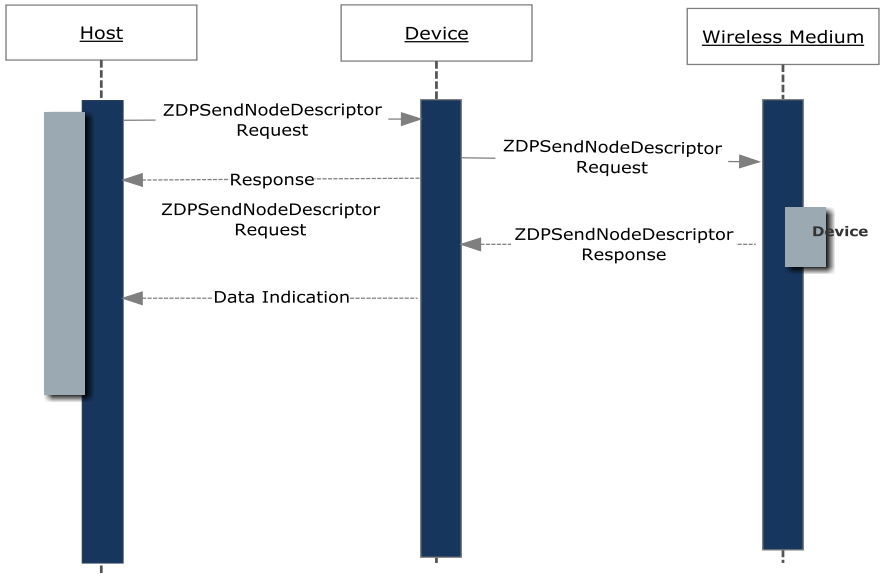

Figure 11: Node Descriptor Request

4.1.56 ZigBeeSimpleDescriptorRequest

Description:

This frame allows the application request to get the simple descriptor of target device..

Supported Modes: End-Device, Router and Coordinator.

Prerequisites: The zigBee stack must be initialized and the device must be in the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESIMPLEDESCRIPTORREQUEST (0x34)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t    ShortAddr;
    uint8_t     EndPointId;
} getSimpleDescOfShortAddrFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
ShortAddr	Interger	0x0000 - 0xffff	The device short address whose matching end-points are desired.

EndPointId	Integer	1-254	The endpoint on the destination.

Table 40 Simple Descriptor Request Parameters

Expected Response Frames:
The following command frames are expected:

1. For this command frame default ZigBeeCommandResp response frame is expected for indicating status and
2. AppHandleDataIndicationResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

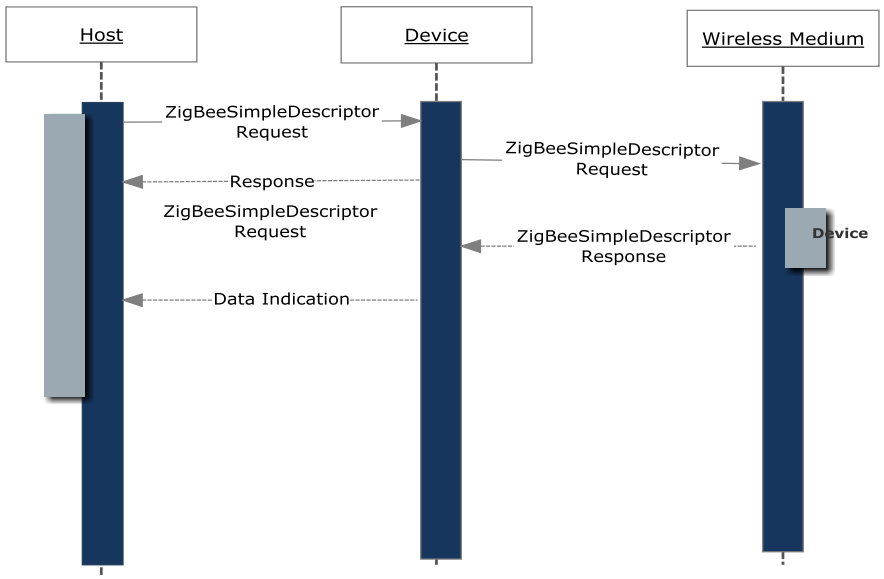

Figure 12: Simple Descriptor Request

4.1.57 ZigBeeInitPS:

Description:

This frame allows the application to read the number of child devices on the node.

Supported Modes: EndDevice.

Prerequisites: The device must be End Device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEINITPS (0x68)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint8_t     ps_en;
    uint8_t     deep_sleep_wkp_period;
    uint8_t     slp_mode;
} pwrModeFrameSnd;
```

Payload Parameters:

3. **ps_en**: Power save Enable/Disable.
   - **ENABLE** : 0x1
   - **DISABLE** : 0x0

4. **Deep_sleep_wkp_period** : the deep sleep wake up period in seconds.

5. **Slp_mode**: **The device sleep mode,**
   - **LP_MODE**: 0x1
   - **ULP_MODE**: 0x2

Expected Response Frames:
For this command frame default ZigBeeCommandResp response frame is expected for indicating status

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.2 Data Frames

4.2.1 ZigBeeSendUnicastData

Description:

This frame allows the application to initiate APSDE data request to the specified destination address.

Supported Modes: End-Device, Router and Coordinator.

Prerequisites: The zigBee stack must be initialized and the device must be in the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESENDUNICASTDATA (0x36)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t    ProfileId;
    uint16_t    ClusterId;
    uint8_t     msgType;
    uint8_t     ieee_Addr[8];
    uint8_t     DestEndpoint;
    uint8_t     SrcEndpoint;
    uint8_t     AsduLength;
    uint8_t     TxOptions;
    uint8_t     Radius;
    uint8_t     Data[200];
} unicastDataFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
ProfileId	Integer	0x0000 – 0xffff	The identifier of the profile for which this frame is intended.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the object for which this frame is intended
msgType	Integer	0x00– 0x04	ZigBee_Outgoing_Direct (0x00) ZigBee_Via_Address_Map (0x01) ZigBee_Via_Binding_Table(0x02) ZigBee_Via_Multicast (0x03) ZigBee_Broadcast (0x04)
IEEE_Addr	IEEE	A valid 64-bit	Address of the

	Address(Integer)	IEEE address	destination device
DestEndpoint	Integer	0x00 – 0xff	This parameter shall be present if, and onlyif, the DstAddrMode parameter has a value of 0x02 or 0x03 and, if present, shall be either the number of the individual endpoint of the entity to which the ASDU is being transferred or the broadcast endpoint (0xff).
SrcEndpoint	Integer	0x00 – 0xfe	The individual endpoint of the entity from which the ASDU is being transferred.
AsduLength	Integer	0x00 - 256*(NsduLength - apscMinHeaderOverhead)	The number of octets comprising the ASDU to be transferred. The maximum length of an individual APS frame payload is given as NsduLength - *apscMinHeaderOverhead*. Assuming fragmentation is used, there can be 256 such blocks comprising a single maximum sized ASDU.
TxOptions	Bitmap	0000 0000 – 00011111	The transmission options for the ASDU to be transferred. These are a bitwise OR of one or more of the following: 0x01 = Security enabled transmission 0x02 = Use NWK key 0x04 = Acknowledged transmission 0x08 = Fragmentation permitted 0x10 = Include extended nonce in APSsecurity frame
Radius	Unsigned integer	0x00-0xff	The distance, in hops, that a transmitted frame will be allowed to travel through the network.
Data	Unsigned integer	0 - 120	The payload

Table 41 Send Uni-cast Data Parameters

Expected Response Frames:
The following command frames are expected:

1. For this command frame default ZigBeeCommandResp response frame is expected for indicating status and

2. AppHandleDataConfirmationResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

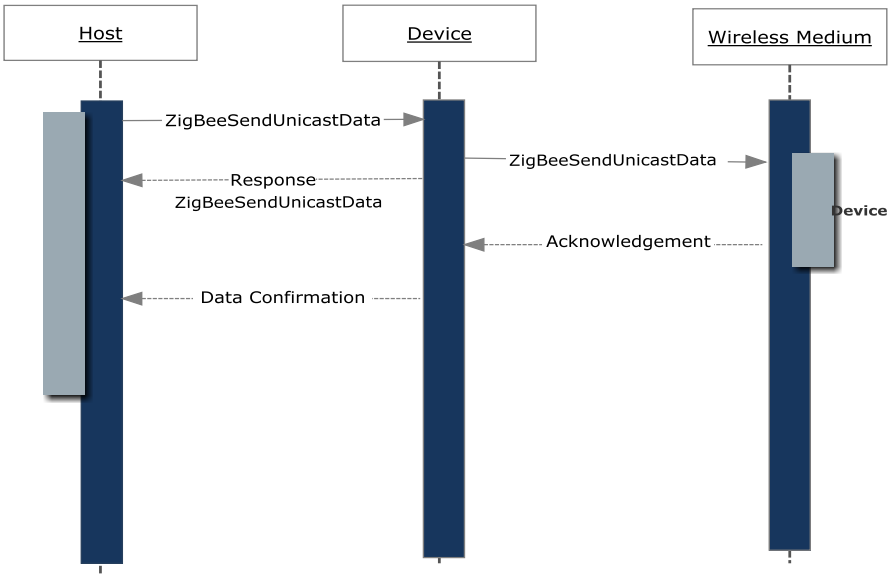

Figure 13: Send Data

4.2.2 ZigBeeSendGroupData

Description:

This frame allows the application to initiate APSDE data request to the specified group address.

Supported Modes: End-Device, Router and Coordinator.

Prerequisites: The zigBee stack must be initialized and the device must be in the network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESENDGROUPDATA (0x37)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

```
struct {
    uint16_t    ProfileId;
    uint16_t    ClusterId;
    uint16_t    group_addr;
    uint8_t     DestEndpoint;
    uint8_t     SrcEndpoint;
    uint8_t     AsduLength;
    uint8_t     TxOptions;
    uint8_t     Radius;
    uint8_t     Data[200];
} groupDataFrameSnd;
```

Payload Parameters:

Name	Type	Valid Range	Description
ProfileId	Integer	0x0000 – 0xffff	The identifier of the profile for which this frame is intended.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the object for which this frame is intended
group_addr	16-bit group address	0x0000 - 0xffff	The 16-bit address of the group being added.
DestEndpoinst	Integer	0x00 –0xff	This parameter shall be present if, and onlyif, the DstAddrMode parameter has a value of 0x02 or 0x03 and, if present, shall be either the number of the individual endpoint of the entity to which the ASDU is being transferred or the broadcast endpoint (0xff).
SrcEndpoint	Integer	0x00 – 0xfe	The individual endpoint of the entity from which the ASDU is being transferred.
AsduLength	Integer	0x00 - 256*(NsduLength - apscMinHeader Overhead)	The number of octets comprising the ASDU to be transferred. The maximum length of an individual APS frame payload is given as NsduLength - *apscMinHeaderOverhead*. Assuming fragmentation is used, there can be 256 such blocks comprising a single maximum sized ASDU.
TxOptions	Bitmap	0000 0000 – 00011111	The transmission options for the ASDU to be transferred. These are a bitwise OR of one or more of the following: 0x01 = Security enabled transmission 0x02 = Use NWK key 0x04 = Acknowledged transmission 0x08 = Fragmentation permitted 0x10 = Include extended nonce in APSsecurity frame
Radius	Unsigned integer	0x00-0xff	The distance, in hops, that a transmitted frame will be allowed to travel through the network.

Data	Unsigned integer	0 - 120	The payload

Table 42 Send Group Data Parameters

Expected Response Frames:
The following command frames are expected:

1. For this command frame default ZigBeeCommandResp response frame is expected for indicating status and

2. AppHandleDataConfirmationResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.2.3 ZigBeeGetMaxAPSPayloadLength

Description:

This frame allows the application to read the maximum size of the payload that the Application Support sub-layer will accept. The size depends on the security level in use. The value is the same as that found in the node descriptor.

Supported Modes: End-Device, Router and Coordinator.

Prerequisites: The zigBee stack must be initialized.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETMAXAPSPAYLOADLENGTH (0x39)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Payload Structure:

Only Descriptor is sent payload is not required for this frame.

Paylaod Parameters: None

Expected Response Frames:
Command responses for this request are ZigBeeGetMaxAPSPayloadLengthResponse and AppHandleDataIndicationResp

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

4.3 Security Frames

4.3.1 ZigBeeGetKey

Description:

This frame allows the Application to get specified key and its associated data. Using this frame user can retrieve Link key, Current Network Key, or Next Network Key.

Supported Modes: End_device, Router and Coordinator

Prerequisites: The zigBee stack must be initialized

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETKEY (0x41)

Interface Type - SECURITY_INTERFACE (0x3)

Payload Structure:

```
struct {
    Security_Key_Types  KeyType;
} getKeyFrameSnd;

typedef enum Security_Key_Types_Tag {
      g_Trust_Center_Master_Key_c,
      g_Network_Key_c,
      g_Application_Master_Key_c,
      g_Link_Key_c,
      g_Trust_Center_Link_Key_c,
      g_Next_Network_Key_c
} Security_Key_Types
```

Paylaod Parameters:

1. **KeyType** : Indicates the type of key sent and waits for the associated response frame. Possible values for KeyType are:

- **g_Trust_Center_Master_Key_c:** Get trust center master key
- **g_Network_Key_c:** Get network key
- **g_Application_Master_Key_c:** Get application master key
- **g_Link_Key_c:** Get link key
- **g_Trust_Center_Link_Key_c:** Get trust center(co-ordinator) link key
- **g_Next_Network_Key_c**: Get next network link key

KeyType	Value
g_Trust_Center_Master_Key_c (Reserved)	**0x0**
g_Network_Key_c	**0x1**
g_Application_Master_Key_c (Reserved)	**0x2**
g_Link_Key_c (Reserved)	**0x3**
g_Trust_Center_Link_Key_c	**0x4**

2.
3.

Table 43 Key Types

Expected Response Frames:

For this command frame ZigBeeGetKeyResp response frame is expected

Note:

Presently only Current Network Key, Next Network Key and Trust Center Link key are supported and other key types are not supported. So when any other key type other than the above specified type is sent then g_Invalid_Request_c will be sent as response.

4.3.2 ZigBeeHaveLinkKey

Description:

This frame allows the Application to get information about trust center link key from our device so that we are securing our messages sent to the partner joined.

Supported Modes: End_device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEHAVELINKKEY (0x42)

Interface Type - SECURITY_INTERFACE (0x3)

Payload Structure:

```
struct {
    uint8_t  ieee_Addr[8];
} haveLinkKeyFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
IEEE_Addr	IEEE Address(Integer)	A valid 64-bit IEEE address	It is the IEEE Addr of the partner connected to.

Table 44 Have Link Key Parameters

Expected Response Frames:

For this command frame ZigBeeGetKeyResp response frame is expected

4.3.3 ZigBeeSwitchNetworkKey

Description:

This frame allows the Application to switch network key. This frame will be sent to device and it request every body to switch current key.

This frame is reserved.

4.3.4 ZigBeeRequestLinkKey

Description:

This frame allows the Application to get trust center link key for the child joined.

Supported Modes: Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEREQUESTLINKKEY (0x44)

Interface Type - SECURITY_INTERFACE (0x3)

Payload Structure:

```
struct {
    uint8_t     TrustCenterIEEEAddr[8];
    uint8_t     PartnerIEEEAddr[8];
} reqLinkKeyFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
TrustCenter IEEEAddr	Device address(Integer)	Any valid 64-bit address	Identifies the address of the device's Trust Center.
Partner IEEEAddr	Device address(Integer)	Any valid 64-bit address	If the KeyType parameter indicates an application key, this parameter shall indicate an extended 64-bit address of a device that shall receive the

			same key as the device requesting the key.

Table 45 Request Link Key Parameters

Expected Response Frames:

For this command frame ZigBeeCommandResp response frame is expected

4.3.5 ZigBeeGetKeyTableEntry

Description:

This frame allows the Application to get the key configuration for the given index.

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETKEYTABLEENTRY (0x45)

Interface Type - SECURITY_INTERFACE (0x3)

Payload Structure:

```
struct {
    uint8_t     Index;
} getKeyTableFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Index	Integer	0x00 – 0xff	Index From Where Key structure information is gathered.

Table 46 Get Key Table Entry Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

4.3.6 ZigBeeSetKeyTableEntry

Description:

This frame allows the Application to set the key configuration for the given index.

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESETKEYTABLEENTRY (0x46)

Interface Type - SECURITY_INTERFACE (0x3)

Payload Structure:

```
struct {
    uint8_t     Index;
    uint8_t     ieee_Addr[8];
    uint8_t     LinkKey;
    uint8_t     KeyData[16];
} setKeyTableFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Index	Integer	0x00-0xff	Index entry where key table information is set.
IEEE_Addr	IEEE Address(Integer)	A valid 64-bit IEEE address	It is the IEEE Addr of the partner connected to.
LinkKey	Set of 16 octets	Variable	The actual value of the link key.if link key type is : 0x00 = Unique Link Key 0x01 = Global Link Key.
KeyData	Integer	Variable	Key to update

Table 47 Set Key Table Entry Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

4.3.7 ZigBeeAddOrUpdateKeyTableEntry

Description:

This frame allows the Application to add a new entry in the key table or update an existing entry with a new key. It first searches the key table for an entry that has a matching IEEE Address. If it does not find one, it searches for the first free entry. If it is successful in either case, it sets

the entry with the IEEE address, key data, and flag that indicates if it is a Link or Master Key. The Incoming Frame Counter for that key is also reset to 0. If no existing entry is found, and there is no free entry in the table, then the call will fail.

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEADDORUPDATEKEYTABLEENTRY (0x47)

Interface Type - SECURITY_INTERFACE (0x3)

Payload Structure:

```
struct {
    uint8_t     ieee_Addr[8];
    uint8_t     LinkKey;
    uint8_t     KeyData[16];
} addKeyTableFrameSnd
```

Paylaod Parameters:

Name	Type	Valid Range	Description
IEEE_Addr	IEEE Address(Integer)	A valid 64-bit IEEE address	It is the IEEE Addr of the partner connected to.
LinkKey	Set of 16 octets	Variable	The actual value of the link key.if link key type is : 0x00 = Unique Link Key 0x01 = Global Link Key.
KeyData	Integer	Variable	Key to update

Table 48 Update Key Table Entry Parameters

Expected Response Frames:

The expected response for this frame would be the index number from the entry table where it is updated. For this command frame ZigBeeAddOrUpdateKeyTableEntryResp response frame is expected

4.3.8 ZigBeeFindKeyTableEntry

Description:

This frame allows the Application to find the table entry using the 64-bit extended address(IEEE Address).

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEFINDKEYTABLEENTRY (0x48)

Interface Type - SECURITY_INTERFACE (0x3)

Payload Structure:

```
struct {
    uint8_t     ieee_Addr[8];
    uint8_t     LinkKey;
} findKeyTableFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
IEEE_Addr	IEEE Address(Integer)	A valid 64-bit IEEE address	It is the IEEE Addr of the partner connected to.
LinkKey	Set of 16 octets	Variable	The actual value of the link key.if link key type is : 0x00 = Unique Link Key 0x01 = Global Link Key.

Table 49 Find Key Table Entry Parameters

Expected Response Frames:

The expected response for this frame would be the index number from the entry table where it is found. For this command frame ZigBeeFindKeyTableEntryResp response frame is expected.

4.3.9 ZigBeeEraseKeyTableEntry

Description:

This frame allows the Application to erase the key table entry based on Index provided

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEERASEKEYTABLEENTRY (0x49)

Interface Type - SECURITY_INTERFACE (0x3)

Payload Structure:

```
struct {
    uint8_t     Index;
} eraseKeyTableFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Index	Integer	0x00-0xff	Index of key table from which key entry has to be erased

Table 50 Earse Key Table Entry Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

4.4 Binding Frames

4.4.1 ZigBeeSetBindingEntry

Description:

This frame allows the Application to set binding entry in ZigBee stack. With this frame the source End point and destination End point info will be stored in the binding table.

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEESETBINDINGENTRY (0x3A)

Interface Type - BINDING_INTERFACE (0x4)

Payload Structure:

```
struct {
    uint8_t     SrcIEEEAddr[8];
    uint8_t     SrcEndpoint;
    uint16_t    ClusterId;
    uint8_t     DestAddrMode;
    uint8_t     DestAddress;/*8 bytes are reserved*/
    uint8_t     DestEndpoint;
} setBindEntryFrameSnd;

union Address_Tag {
    uint16_t short_address;
    uint8_t IEEE_address[8];
} Address;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
SrcIEEEAddr	IEEE Address	A valid 64-bit IEEE address	The IEEE address for the source.
SrcEndpoint	Integer	0x01-0xfe	The source endpoint for the binding entry.
ClusterId	Integer	0x0000-0xffff	The identifier of the cluster on the source device that is bound to the destination.
DestAddrMode	Integer	0x00-0xff	The addressing mode for the destination address used in this command. This field can take

			one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 – 0xff = reserved
DestAddress	Address	As specified by the DstAddrMode field	The destination address for the binding entry.
DestEndpoint	Integer	0x01-0xfe	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

Table 51 Set Binding Entry Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

4.4.2 ZigBeeGetBindingIndices

Description:

This frame allows the Application to get the active binding indices from ZigBee stack.

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEGETBINDINGINDICES (0x60)

Interface Type - BINDING_INTERFACE (0x4)

Payload Structure: None.
Paylaod Parameters: None.

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

4.4.3 ZigBeeDeleteBinding

Description:

This frame allows the Application to delete the binding entry of specified index from binding table.

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEDELETEBINDING (0x3B)

Interface Type - BINDING_INTERFACE (0x4)

Payload Structure:

```
struct {
    uint8_t     BindIndex;
} delBindEntryFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Index	Integer	0x00-0xff	Binding entry of binding table which is to be removed.

Table 52 Delete Binding Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

4.4.4 ZigBeeIsBindingEntryActive

Description:

This frame allows the Application to verify if the binding entry is active or not.

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEISBINDINGENTRYACTIVE (0x3C)

Interface Type - BINDING_INTERFACE (0x4)

Payload Structure:

```
struct {
    uint8_t     BindIndex;
```

```
} isBindEntryActiveFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Index	Integer	0x00-0xff	Binding entry of binding table which is to be verified.

Table 53 Binding Entry Active Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

4.4.5 ZigBeeClearBindingTable

Description:

This frame allows the Application to clear the formed binding table.

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEECLEARBINDINGTABLE (0x3D)

Interface Type - BINDING_INTERFACE (0x4)

Payload Structure:

Only Descriptor is sent and payload is not required for this frame.

Paylaod Parameters:

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

4.4.6 ZigBeeBindRequest

Description:

This frame allows the Application to bind source and destination endpoints. With this frame the source End point and destination End point will be linked logically which can be later used for data communication.

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEBINDREQUEST (0x3E)

Interface Type - BINDING_INTERFACE (0x4)

Payload Structure:

```
struct {
    uint16_t    SrcShortAddr;
    uint16_t    ClusterId;
    uint8_t     SrcIEEEAddr[8];
    uint8_t     SrcEndpoint;
    uint8_t     DestAddrMode;
    Address     DestAddress;/*8 bytes are reserved*/
    uint8_t     DestEndpoint;
    uint8_t     APSAckRequired;
} bindReqFrameSnd;

union Address_Tag {
    uint16_t short_address;
    uint8_t IEEE_address[8];
};
```

Payload Parameters:

Name	Type	Valid Range	Description
SrcShortAddr	Interger	0x0000 - 0xffff	The device short address .
ClusterId	Integer	0x0000-0xffff	The identifier of the cluster on the source device that is bound to the destination.
SrcIEEEAddr	IEEE Address	A valid 64-bit IEEE address	The IEEE address for the source.
SrcEndpoint	Integer	0x01-0xfe	The source endpoint for the binding entry.
DestAddrMode	Integer	0x00-0xff	The addressing mode for the destination address used in this command. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 – 0xff = reserved
DestAddress	Address	As specified by the DstAddrMode field	The destination address for the binding entry.
DestEndpoint	Integer	0x01-0xfe	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

1. **SrcShortAddr**: Short address of

Table 54 Bnd Request Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

4.4.7 ZigBeeUnBindRequest

Description:

This frame allows the Application to unbind source and destination endpoints. With this frame logical link between Source End point and Destination End point will be terminated.

Supported Modes: End_Device, Router and Coordinator

Prerequisites: The device must be part of a network or a network is formed by device.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEUNBINDREQUEST (0x40)

Interface Type - BINDING_INTERFACE (0x4)

Payload Structure:

```
struct {
    uint16_t    SrcShortAddr;
    uint16_t    ClusterId;
    uint8_t     SrcIEEEAddr[8];
    uint8_t     SrcEndpoint;
    uint8_t     DestAddrMode;
    Address     DestAddress;/*8 bytes are reserved*/
    uint8_t     DestEndpoint;
    uint8_t     APSAckRequired;
} unbindReqFrameSnd;

union Address_Tag {
    uint16_t short_address;
    uint8_t IEEE_address[8];
} Address;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
SrcShortAddr	Interger	0x0000 - 0xffff	The device short address .
ClusterId	Integer	0x0000-0xffff	The identifier of the cluster on the source device that is bound to the destination.
SrcIEEEAddr	IEEE Address	A valid 64-bit IEEE address	The IEEE address for the source.
SrcEndpoint	Integer	0x01-0xfe	The source endpoint for the binding entry.
DestAddrMode	Integer	0x00-0xff	The addressing mode for the destination address used in this command. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 – 0xff = reserved
DestAddress	Address	As specified by the DstAddrMode field	The destination address for the binding entry.
DestEndpoint	Integer	0x01-0xfe	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

Expected Response Frame for Unbind Request Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status

4.4.8 ZigBeeEndDeviceBindRequest

Description:

To bind two end devices of same network then this frame should be used from host application. Both the end devices should send bind request simultaneously to Co-ordinator, if Co-ordinator receives end device bind

request first then it will wait for timeout duration to receive other bind request from other end device. Later these to end devices will be binded logically by Co-ordinator.

By default the EndDevice Binding request timeout is configured to 10 seconds.

Supported Modes: End_Device

Prerequisites: The device must be part of a network.

Descriptor: 16 bytes Frame Descriptor should be sent first then payload should be followed by descriptor.

Command Type - ZIGBEEENDDEVICEBINDREQUEST (0x3F)

Interface Type - BINDING_INTERFACE (0x4)

Payload Structure:

```
struct {
    uint8_t     EndPointId;
    uint8_t     APSAckRequired;
} endDevBindFrameSnd;
```

Paylaod Parameters:

Name	Type	Valid Range	Description
Endpoint Id	8 bits	1-254	The endpoint on the device generating the request
APSAckRequired	Interger	0x00 – 0x01	TRUE (0x00) indicates APS ack is required.

Table 56 End device Bind Request Parameters

Expected Response Frames:

For this command frame default ZigBeeCommandResp response frame is expected for indicating status.

5 Response frames

This section contains description of various response frames received from device. All the response frames contains 4-byte length descriptor, reserved dummy bytes, 16-byte descriptor and payload as described in Rx Operation. The 16-byte descriptor will be similar for all the received command frames. The fields which differ for each command descriptor are length of payload, Interface type and Command type. In the descriptor direction would be from device to host.

5.1 Default Status Frame

5.1.1 ZigBeeCommandResp

Description:

Once device receives command request, it validates the input parameters and sends the common response frame to host for most of the command requests with status success or failure or invalid etc.,. Based on the type of command additional payload may be expected for few frames, but for this response frame status is only expected.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This generic response is expected when a command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t  status;
} rsi_StatusResp;
```

Payload Parameters:

The response payload structure may contain one of the status as specified in the table Zigbee status Codes (Error Codes).

5.2 Event Callbacks

5.2.1 ZigBeeCardReady

Description:

This is the first packet received from device, intimating that the device is ready to accept commands. This will also come as a response for de-init command.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

The command type is 0xFF for card ready and interface type is 0x6(EVENTCALLBACKS).

5.2.2 AppNetworkFoundHandlerResp

Description:

This event callback is triggered from the stack(device) to inform about the networks found in the current channel. The network information is beacon content.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This frame is expected when ZigBeeInitiaiteScan command frame is sent to device from host and if the device detects any new networks in the channels specified in Channel Mask**Payload Structure:**

```
Struct {
      uint16_t    shortPanId,
      uint8_t     channel,
      uint8_t    extendedPanId[8],
      uint8_t    stackProfile,
      uint8_t    nwkUpdateId
      bool       allowingJoining,
} ZigBeeNetworkDetails;
```

Payload Parameters:

1. **channel:** The 802.15.4 channel associated with the network.
1. **ShortPanId** : The network's PAN identifier.
2. **extendedPanId**[8] : The network's extended PAN identifier.
3. **allowingJoining** : Whether the network is allowing MAC associations.
4. **stackProfile** : The Stack Profile associated with the network.
5. **nwkUpdateId** : The instance of the Network.

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

5.2.3 AppScanCompleteHandlerResp

Description:

This event callback is triggered from the stack(device) to inform the status of the current channel scan to the application.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This is expected when ZigBeeInitiaiteScan command frame is sent to device from host and when the scan for the specified duration is completed.

Payload Structure:

```
Struct {
```

```
    uint32_t    channel,
    uint32_t    scan_status,
}rsi_ScanDoneResp;
```

Payload Parameters:

1. **Channel:** The channel on which the scan occurred

2. **Scan_status:** M**ac status obtained would be one of the below specified status**

MAC Scan Status	Value
g_MAC_Success_c	**0x0**
g_PAN_At_Capacity_c	0x1
g_PAN_Access_denied_c	0x2
g_MAC_Scan_In_Progress_c	0xAA
g_MAC_Beacon_Loss_c	0xE0
g_MAC_Channel_Access_Failure_c	0xE1
g_MAC_Denied_c	0xE2
g_MAC_Disable_TRX_Failure_c	0xE3
g_MAC_Failed_Security_Check_c	0xE4
g_MAC_Frame_Too_Long_c	0xE5
g_MAC_Invalid_GTS_c	0xE6
g_MAC_Invalid_Handle_c	0xE7
g_MAC_Invalid_Parameter_c	0xE8
g_MAC_No_ACK_c	0xE9
g_MAC_No_Beacon_c	0xEA
g_MAC_No_Data_c	0xEB
g_MAC_No_Short_Address_c	0xEC
g_MAC_Out_Of_CAP_c	0xED
g_MAC_PAN_ID_Conflict_c	0xEE
g_MAC_Realignment_c	0xEF
g_MAC_Transaction_Expired_c	0xF0
g_MAC_Transaction_Overflow_c	0xF1
g_MAC_TX_Active_c	0xF2
g_MAC_Unavailable_Key_c	**0xF3**
g_MAC_Unsupported_Attribute_c	**0xF4**
g_MAC_Missing_Address_c	**0xF5**

g_MAC_Past_Time_c	**0xF6**

Table 57 MAC Scan status Types

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

5.2.4 AppEnergyCompleteHandlerResp

Description:

This event callback is triggered from the stack(device) to report RSSI value measured in the required channel.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This is expected when ZigBeeInitiaiteScan command frame is sent to device from host and when the energy scan for the specified duration is completed.

Payload Structure:

```
Struct{
      uint32_t    channel,
      uint8_t     PEnergyValues[16],
}rsi_EnergyScanDoneResp;
```

Payload Parameters:

1. **Channel:** The channel on which the scan occurred.
2. **pEnergyValues[16] :** This array holds the RSSI values for the channels from 11 to 26.

API: For more information about frame usage, refer corresponding API in sample application. Refer Appendix for Command type and API name.

5.2.5 AppHandleDataConfirmationResp

Description:

This event callback is data confirmation response for the data, which is sent to device from host. This event callback is triggered for every data packet sent over the air.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This response is expected when data is transmitted from the device over the air.

Payload Structure:

```
struct{
    Address dest_address;
    uint8_t dest_addr_mode;
    uint8_t dest_endpoint;
    uint8_t src_endpoint;
    uint8_t status;
}APSDE_Data_Confirmation_t;

union{
uint16_t short_address;
uint8_t IEEE_address[8];
```

```
}Address;
```

Payload Parameters:

1. dest_address: This field indicates the individual device address or group address of the transmitted message.

- **short_address** : The short address of the device.
- **IEEE_address** : The IEEE address of the device.

2. **dest_addr_mode :**

   This field indicates the destination address mode. This field takes one of the following values:

   - 0x00 - Indirect data transmission (destination address and destination endpoint are not present)
   - 0x01 - 16-bit group address
   - 0x02 - 16-bit address of destination device
   - 0x03 - 64-bit extended address of destination device
   - 0x04 - 0xff - Reserved

3. **dest_endpoint :** This field indicates the destination endpoint to which the data frame was sent.
4. **src_endpoint :** This field indicates the source endpoint from which the data frame was originated.
5. **Status :** This field indicates the status of data confirmation. For details of the various status values refer Zigbee status Codes.

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

5.2.6 AppHandleDataIndicationResp

Description:

This event callback is triggered from the stack (device) to inform that data is pending for reading. This is a data indication frame sent to inform that data is received by device for the data request sent.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This frame is expected when data request is transmitted from the device on air.

Payload Structure:

```
struct {
    Address     dest_address;
    uint8_t     dest_addr_mode;
    uint8_t     dest_endpoint;
    uint8_t     src_addr_mode;
    Address     src_address;
```

```
    uint8_t     src_endpoint;
    profile_id_t profile_id;
    cluster_id_t cluster_id;
    uint8_t     asdulength;
    uint8_t     was_broadcast;
    uint8_t     security_status;
    uint8_t     link_quality;
    uint8_t     a_asdu[1];
} APSDE_Data_Indication_t;

union{
    uint16_t short_address;
    uint8_t IEEE_address[8];
}Address;
```

Payload Parameters:

1. dest_address: This field the destination address in the received message.

- **short_address** : The short address of the device.
- **IEEE_address** : The IEEE address of the device.

2. **dest_addr_mode :** This field indicates the destination address mode in the receivedmessage. This field takes one of the following values:

- 0x00 - Indirect data transmission (destination address and destination endpoint are not present)
- 0x01 - 16-bit group address
- 0x02 - 16-bit address of destination device
- 0x03 - 64-bit extended address of destination device
- 0x04 - 0xff - Reserved

3. dest_endpoint : This field indicates the destination endpoint in the received message.

4. **src_addr_mode :** This field indicates the source address mode in the received message. This field can have one of the following values:

- 0x00 - Indirect data transmission (destination address and destination endpoint are not present)
- 0x01 - 16-bit group address
- 0x02 - 16-bit address of destination device
- 0x03 - 64-bit extended address of destination device
- 0x04 - 0xff - Reserved

5. **src_address :** This field indicates the source address from which the message is originated. This field can have one of the following values:

- If the source address mode is 0x01, this field will have 16-bit

address.

- If source address mode is 0x03, this field will have 64-bit

extended address.

6. **profile_id:** This field indicates the 16-bit profile ID**.**

7. **cluster_id -** This field indicates the cluster ID**.**

8. **Asdulength -** This field indicates the length of the data received.

9. **was_broadcast** - This field indicates whether the data frame is received through broadcast.

10. **security_status :** This field indicates whether the received message was secured or not and type of the security applied. The enum values are as follows:

- **g_APS_UNSECURED_c - ASDU is received without any security.**
- **g_Recieved_Nwk_Key_Secured_Asdu_c - ASDU is received**

security using the Network Key.

- **g_Recieved_Link_Key_Secured_Asdu_c - ASDU is received with security using the link Key.**

11. **link_quality**: **This field indicates the LQI of the received message.**

12. **a_asdu[1]**: This field points to the actual message received.

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

5.2.7 AppChildJoinResp

Description:

This event callback is triggered from the stack(device) to intimate about child device joining or leaving the network.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This is expected when a child joins/leaves the device.
Payload Structure:

```
Struct{
    uint16_t          short_addr,
    uint8_t           joined,
}rsi_EnergyScanDoneResp;
```

Payload Parameters:

1. **short_**addr**:** The child's short address.

2. **joined :**

- TRUE : Child joined the device.

- FALSE : Child left the network.

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

5.2.8 AppIncomingManyToOneRouteResp

Description:

This event callback is triggered from the stack(device) to handle many to One Route Request.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This is expected when many to one route request is initiated.
Payload Structure:

```
Struct{
    uint16_t          source_addr,
    uint8_t           source_ieee[8],
    uint8_t           Cost;
}rsi_EnergyScanDoneResp;
```

Payload Parameters:

1. **source_addr:** The short address of the concentrator which initiated the many to one request.
2. **source_ieee:** The concentrator's ieee address.
3. **Cost:** The path cost of the concentrator.

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

5.2.9 AppZigBeeStackStatusHandlerResp

Description:

This event callback is triggered from the stack(device) to indicate any kind of Network status.
For example: Upon establishing the network, this function shall be called by the stack to indicate status ZigBeeNetworkIsUp. If the device leaves the network, a status of ZigBeeNWkisDown status is indicated via this event callback.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This is expected when ZigBeeJoinNetwork or ZigBeeFormNetwork command frame is sent to device from host or when the device leaves/joins the network or when network is demolished, this event callback will be triggered

Payload Structure:

```
enum {
    ZigBeeNWKIsUp,
    ZigBeeNWKIsDown,
    ZigBeeJoinFailed,
    ZigBeeCannotJoinAsRouter,
    ZigBeeChangedNodeID,
    ZigBeeChangedNodeID,
    ZigBeeChangedChannel,
    ZigBeeNoBeacons,
    ZigBeeReceivedKeyInClear,
    ZigBeeNoNWKKeyReceived,
    ZigBeeNoLinkKeyReceived,
    ZigBeePreconfiguredKeyRequired,
    ZigBeeChangedManagerAddress
} ZigBeeNWKStatusInfo;
```

Payload Parameters:

1. **ZigBeeNWKIsUp** indicates that Network is formed or joined successfully.
2. **ZigBeeNWKIsDown** indicates that NWK formation failed or the device left the network.
3. **ZigBeeJoinFailed** indicates that network join failed.
4. **ZigBeeCannotJoinAsRouter** indicates that network was unable to start as Router.
5. **ZigBeeChangedNodeID** indicates that PANID is changed after resolving PAN ID conflict.
6. **ZigBeeChangedChannel** indicates that the channel is changed due to frequency agility mechanism
7. **ZigBeeReceivedKeyInClear** indicates the Network Key is received is inclear.
8. **ZigBeeNoNWKKeyReceived** indicates no Network key is received.
9. **ZigBeeNoLinkKeyReceived** indicates no Link key is received.
10. **ZigBeePreconfiguredKeyRequired** indicates Preconfigured link key is required.
11. **ZigBeeChangedManagerAddress** indicates network manager changed.

API: For more information about frame usage refer corresponding API in sample application. Refer Appendix for Command type and API name.

5.3 Other Responses

5.3.1 ZigBeeGetNeighborTableEntryCountResp

Description:

Once device receives a neighbor tableEntry count request, it validates the input parameters and sends the response frame, which contains the status of the request and the neighbor table entry.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEREADNEIGHBORTABLEENTRY (0x1E)

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when ZigBeeGetNeighborTableEntryCount command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t  status;
    ZigBeeNeighborTableEntry_t  Neighbor_table_entry;
}rsi_GetNeighborTableEntryResp;

struct {
    uint16_t         shortId;
    uint8_t          averageLqi;
    uint8_t          incomingCost;
    uint8_t          outgoingCost;
    uint8_t          age;
    uint8_t          aIEEEAddress[8];
}ZigBeeNeighborTableEntry_t;
```

Payload Parameters:

1. **status** can have two values

   - **g_SUCCESS_c**: indicating scan begun successfully.
   - **ZigBee_Invalid_Argument**: indicates invalid arguments being passed to the transmit frame.

2. **Neighbor_table_entry**

   - **shorted** : The neighbor's two byte short address.
   - **averageLqi** : An exponentially weighted moving average of the link quality values of incoming packets from this neighbor as reported by the PHY.
   - **incomingCost** :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.
   - **outgoingCost**: The outgoing cost for this neighbor, obtained from the most recently received neighbor exchange message from the neighbor.
   - **Age** : The number of aging periods elapsed since a neighbor exchange message was last received from this neighbor. An entry with an age greater than 3 is considered stale and may be reclaimed. The aging period is 16 seconds.
   - **aIEEEAddress** : The 8 byte IEEE address of the neighbor.

5.3.2 ZigBeeGetChildShortAddressForTheIndexResp

Description:

On device receives ZigBeeGetChildShortAddressForTheIndex frame, then the device sends 16-bit short address of the child in the specified index if found else INVALID_ADDRESS.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETCHILDINDEXFORSPECIFIEDSHORTADDR(0x22)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeGetChildShortAddressForTheIndex command is sent to device from host.

Payload Structure:

```
struct {
    uint16_t  short_addr;
} rsi_ShortAddrResp;
```

Payload Parameters:

1. **short_addr**
   - 16-bit short address of the child on success,otherwise
   - INVALID_ADDRESS = 0xFFFF.

5.3.3 ZigBeeInitiaiteScanResp

Description:

Once device receives scan request, it validates the input parameters and sends the response frame, which contains the status of the scan request.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This is expected when corresponding command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t  status;
} rsi_StatusResp;
```

Payload Parameters:

The response payload structure may contain one of the status as specified in the table Error Codes.

5.3.4 ZigBeeNetworkStateResp

Once device receives ZigBeeNetworkState request, it verifies network state and sends the status of network.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEENETWORKSTATE (0x0A)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeNetworkState command is sent to device from host.

Payload Structure:

```
enum {
    g_ZigBeeNotPartOfNWK_c,
    g_ZigBeeInTheProcessOfJoiningNWK_c,
    g_ZigBeeJoinedNWK_c,
    g_ZigBeeJoinedNWKNoParent_c,
    g_ZigBeePerformingLeaveFromNWK_c
} ZigBeeJoinStatus;
```

Payload Parameters:

1. **g_ZigBeeNotPartOfNWK_c** : indicates the device is not part of any network
2. **g_ZigBeeInTheProcessOfJoiningNWK_c** - indicates the device is in the process of Joining the network
3. **g_ZigBeeJoinedNWK_c** - indicates the device has joined the Network
4. **g_ZigBeeJoinedNWKNoParent_c** - indicates the device has joined the Network, but parent communication has failed
5. **g_ZigBeePerformingLeaveFromNWK_c** - indicates the device is in the process of leaving the Network

5.3.5 ZigBeeGetSelfIEEEAddressResp

Description:

Once device receives ZigBeeGetSelfIEEEAddress request, it sends the IEEE address of device

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETSELFIEEEADDRESS (0x0C).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when ZigBeeGetSelfIEEEAddress command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t    Self_ieee[8];
} rsi_getSelfIEEEAddrResp;
```

Payload Parameters:

1. **Self_ieee**: it holds the 64 bit IEEE address.

5.3.6 ZigBeeGetSelfShortAddressResp

Once device receives ZigBeeGetSelfShortAddress request, it sends short address of device after it has joined/formed network.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETSELFSHORTADDRESS (0x0E).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when ZigBeeGetSelfShortAddress command is sent to device from host.

Payload Structure:

```
struct {
    uint16_t    short_addr;
} rsi_getSelfShortAddrResp;
```

Payload Parameters:

1. **short_addr** : It indicates the 16-bit short address of the device.
   - If Short addr is **(0xFF),** then **it represents that status is failure while retrieving short address**

5.3.7 ZigBeeGetDeviceTypeResp

Once device receives ZigBeeGetDeviceType request, it gathers information about the device being used and sends the device type as response to the command.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETDEVICETYPE (0x14).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeGetDeviceType command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t      status;
    uint8_t      type;
} rsi_DevTypeResp;
```

Payload Parameters:

1. The status can have four values
   - **g_SUCCESS_c:** indicating GetDeviceType successfully.
   - **g_FAILURE_c:** indicating GetDeviceType successfully.

2. The type can have three values
   - "0" for Coordinator

- "1" for Router
- "2" for End device

5.3.8 ZigBeeGetOperatingChannelResp

Once device receives ZigBeeGetOperatingChannel request, it sends the current operating channel number as response.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETOPERATINGCHANNEL (0x15).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when ZigBeeGetOperatingChannel command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t     channel;
}rsi_ChannelResp;
```

Payload Parameters:

1. **channel**: It indicates Current radio channel.

5.3.9 ZigBeeGetShortPANIdResp

Once device receives ZigBeeGetShortPANId request, it sends the response frame which contains pan id of network.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETSHORTPANID (0x16).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when ZigBeeGetShortPANId command is sent to device from host.

Payload Structure:

```
struct {
    uint16_t    pan_id;
}rsi_PanIdResp;
```

Payload Parameters:

1. **pan_id:** It indicates Short PANID of the network
   - If the network is not formed **0xFFFF** is the value returned

5.3.10 ZigBeeGetExtendedPanIdResp

Once device receives ZigBeeGetExtendedPanId request, it sends the response frame which contains Extended Pan Id of device.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETEXTENDEDPANID (0x17).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when ZigBeeGetExtendedPanId command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t    ExtPanId[8];
}rsi_ExtPanIdResp;
```

Payload Parameters:

1. **Ext_PanId:** it indicates networks extended PANID.

5.3.11 ZigBeeGetEndpointIdResp

Once device receives ZigBeeGetEndpointId request, it validates the input parameters and sends the response frame, which contains the status of the scan request.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETENDPOINTID (0x18).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when corresponding command is sent to device from host.

Payload Structure:

```
struct {
    uint16_t    EndPointId;
} rsi_EndPointId;
```

Payload Parameters:

1. **EndPointId:** The valid Endpoint ID located in the specified index.
   - If EndPointId value is (0xF1) it indicates Endpoint value is not valid

5.3.12 ZigBeeGetSimpleDescriptorResp

Once device receives form nerwork request, it validates the input parameters and sends the response frame, which contains the status of the scan request.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETSIMPLEDESCRIPTOR (0x19).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when corresponding command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t     EndPointId;
    uint16_t    ProfileId;
    uint16_t    DevId;
    uint8_t     DevVersion;
    uint8_t     InClusterCnt;
    uint8_t     *InClusterInfo; //Pointer
    uint8_t     OutClusterCnt;
    uint8_t     *OutClusterInfo; //Pointer
}rsi_GetSimpleDescResp;
```

Payload Parameters:

1. **EndPointId:** The Endpoint on which these clusters are defined
2. **ProfileId** : The application profile id
3. **DevId**: Device ID
4. **Device Version**: Device version info
5. **InClusterCnt** : Number of input clusters.
6. **InCluserInfo**: **Input Cluster information buffer indicating various supported input clusters**
7. **OutClusterCnt** : Number of output clusters
8. **OutCluserInfo**: **Output Cluster information buffer indicating various supported output clusters**
9. **ClusterInfo**: Buffer for input and output cluster, supports 20 in and 20 out clusters per endpoint

5.3.13 ZigBeeGetEndpointClusterResp

Once device receives ZigBeeGetEndpointCluster request, it validates the input parameters and sends the response frame, which contains Cluster Id.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETENDPOINTCLUSTER (0x1A).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when corresponding command is sent to device from host.

Payload Structure:

```
struct {
    uint16_t     ClusterId;
}rsi_ClusterResp;
```

Payload Parameters:

1. **ClusterId:** Cluster id of the endpoint's simple descriptor located at the specified index.

   - If **Cluster Id is 0xFFFF** then it represents that it has received invalid parameters

5.3.14 ZigBeeGetShortAddrForSpecifiedIEEEAddrResp

Once device receives ZigBeeGetShortAddrForSpecifiedIEEEAddr request, it validates the input parameters and sends the response frame, which contains Short Address of device.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type – ZIGBEEGETSHORTADDRFORSPECIFIEDIEEEADDR(0x1B)

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when ZigBeeGetShortAddrForSpecifiedIEEEAddr command is sent to device from host.

Payload Structure:

```
struct {
    uint16_t     short_addr;
}rsi_ShortAddrResp;
```

Payload Parameters:

1. **short_addr :** 16-bit short address of the corresponding 64-bit IEEE address if the address is known. INVALID_SHORT_ADDR**(0xFFFF)** is sent if it is not valid or Short address is not assigned or IEEE address is in correct.

5.3.15 ZigBeeStackProfileResp

Once device receives ZigBeeStackProfile request, it validates the input parameters and sends the response frame, which contains stack profile info

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEESTACKPROFILE (0x1C).

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeStackProfile command is sent to device from host.

Payload Structure:

```
struct {
    uint16_t    status;
} rsi_StatusResp;
```

Payload Parameters:

The status can have two values

1. If **Status is 0x01,** stack follows Zigbee standard Profile.
2. If **Status is 0x02,** stack follows Zigbee-Pro standard Profile.

5.3.16 ZigBeeGetIEEEAddrForSpecifiedShortAddrResp

Once device receives ZigBeeGetIEEEAddrForSpecifiedShortAddr request, it validates the input parameters and sends the response frame, which contains the status and ieee address of the device.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type – ZIGBEEGETIEEEADDRFORSPECIFIEDSHORTADDR(0x1D)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeGetIEEEAddrForSpecifiedShortAddr command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t     status;
    uint8_t     IEEE_Addr[8];
} rsi_GetIEEEAddrForShrtAddr;
```

Payload Parameters:

1. **Status:** The status can be of one of the following .
   - **ZigBee_Success(0x00)** on success,else
   - **ZigBee_Failure**

2. **IEEE_Addr[8]:** The IEEE address corresponding to the short address.

5.3.17 ZigBeeReadNeighborTableEntryResp

Once device receives ZigBeeReadNeighborTableEntry request, it validates the input parameters and sends the response frame, which contains the status of the scan request.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEREADNEIGHBORTABLEENTRY (0x1E)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeReadNeighborTableEntry command is sent to device from host.

Payload Structure:

```
struct {
      uint8_t  status;
      ZigBeeNeighborTableEntry_t  Neighbor_table_entry;
  }rsi_StatusResp;

  struct {
      uint16_t         shortId;
      uint8_t          averageLqi;
      uint8_t          incomingCost;
      uint8_t          outgoingCost;
      uint8_t          age;
      uint8_t          aIEEEAddress[8];
  }ZigBeeNeighborTableEntry_t;
```

Payload Parameters:

1. status can have two values

- **g_SUCCESS_c:** indicating scan begun successfully.
- ZigBee_Invalid_Argument: indicates invalid arguments being passed to the transmit frame.

2. Neighbor_table_entry

- **shorted:** The neighbor's two byte short address.
- **averageLqi**: An exponentially weighted moving average of the link quality values of incoming packets from this neighbor as reported by the PHY.
- **incomingCost**: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.
- **outgoingCost**: The outgoing cost for this neighbor, obtained from the most recently received neighbor exchange message from the neighbor.
- **Age**: The number of aging periods elapsed since a neighbor exchange message was last received from this neighbor. An entry with an age greater than 3 is considered stale and may be reclaimed. The aging period is 16 seconds.

- **aIEEEAddress**: The 8 byte IEEE address of the neighbor.

5.3.18 ZigBeeGetRouteTableEntryResp

Once device receives ZigBeeGetRouteTableEntry request, it validates the input parameters and sends the response frame, which contains the status of request and the route table entry.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETROUTETABLEENTRY (0x1F)

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when ZigBeeGetRouteTableEntry command is sent to device from host.

Payload Structure:

```
struct {
     uint8_t      status;
     uint8_t      destAddr;
     uint8_t      nextHop;
     uint8_t      entry_status;
     uint8_t      age;
     uint8_t      concentratorType;
     uint8_t      routeRecordState;              }
    }rsi_StatusResp;
```

Payload Parameters:

1) The status can have three values.

   - ZigBee_Success**(**0x00**) :** No error occured while parsing the required API parameters .
   - ZigBee_Index_Out_Of_Range**(**0x12**) :** Accessing entry is out of range in the table.
   - ZigBee_Invalid_Argument**(**0x06**) :** Argument passed for API is invalid .

2) destAddr: The short id of the destination.

3) nextHop: The short address of the next hop to this destination.

4) entry_status: Indicates whether this entry is active (0), being discovered (1), or unused (0x3)

5) age: The number of seconds since this route entry was last used to send a packet.

6) concentratorType: Indicates whether this destination is a High RAM Concentrator (2), a Low RAM Concentrator (1), or not a concentrator (0).

7) **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.**

5.3.19 ZigBeeTreeDepthResp

Once device receives ZigBeeTreeDepth request, it validates the input parameters and sends the response frame, which contains the status of the scan request.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEETREEDEPTH (0x20).

Interface Type - MANAGEMENT_INTERFACE (0x1).

Expected: This is expected when ZigBeeTreeDepth command is sent to device from host.

Payload Structure:

```
struct {
     uint16_t   tree_depth;
} rsi_TreeDepthResp;
```

Payload Parameters:

1. **tree_depth**: The current tree depth where the device has joined .

5.3.20 ZigBeeGetChildIndexForSpecifiedShortAddrResp

Description:

On reception of ZigBeeGetChildIndexForSpecifiedShortAddr by device, the device sends the index of the child address else the status received is INVALID short address.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETCHILDINDEXFORSPECIFIEDSHORTADDR(0x22)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeGetChildIndexForSpecifiedShortAddr command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t           Index;
} getChildDetailsFrameSnd;
```

Payload Parameters:

1. **Index**
   - index of the child-address on success,otherwise
   - INVALID_Index = 0xFF.

5.3.21 ZigBeeGetChildDetailsResp

Description:

Once device receives request for child details, the device sends child details.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETCHILDDETAILS (0x24)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeGetChildDetails command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t                 status;
    uint8_t                 Ieee_Addr[8];
    uint8_t                 device_type;
} rsi_ChildDetailsResp;
```

Payload Parameters:

1. **status** : The status can have the following values: Refer Zigbee status Codes for values.
   - ZigBee_Success : If the index is valid.
   - ZigBee_Invalid_Argument : If the paramters are wrong.
   - ZigBee_No_Entry : Child is not connected.
2. **Ieee_Addr :** The Ieee address of the child.
3. **device_type :** The device type of child.

5.3.22 ZigBeeReadCountOfChildDevicesResp

Description:

Once device receives read count device request, it sends the response frame, which contains the number of child devices joined.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEREADCOUNTOFCHILDDEVICES (0x26)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeReadCountOfChildDevices command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t     child_count;
} rsi_CountOfChildResp;
```

Payload Parameters:

1. **child_count** : Number of child devices joined.

5.3.23 ZigBeeReadCountOfRouterChildDevicesResp

Description:

Once device receives read count device request, it sends the response frame, which contains the number of child devices joined.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEREADCOUNTOFROUTERCHILDDEVICES (0x27)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeReadCountOfRouterChildDevices command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t     child_count;
} rsi_CountOfRouterChildResp;
```

Payload Parameters:

1. **child_count** : Number of child devices joined.

5.3.24 ZigBeeGetParentShortAddressResp

Description:

Once device receives parent short address request, it sends the response frame, which contains the 16-bit short address of the parent.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETPARENTSHORTADDRESS (0x29)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeGetParentShortAddress command is sent to device from host.

Payload Structure:

```
struct {
    uint16_t          short_addr;
} rsi_ShortAddrResp;
```

Payload Parameters:

1. **short_addr** :
   - short address of the parent ,
   - Invalid Address : 0xFFFF,

5.3.25 ZigBeeGetParentIEEEAddressResp

Description:

Once device receives parent ieee address request, it sends the response frame, which contains the 64-bit ieee address of the parent.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETPARENTIEEEADDRESS (0x2A)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeGetParentIEEEAddress. command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t           Self_ieee[8];
} rsi_SelfIEEEAddrResp;
```

Payload Parameters:

1. Self_ieee :
   - Ieee Address of the parent.

5.3.26 ZigBeeGetMaxAPSPayloadLengthResp

Description:

Once device receives request to read the maximum APS payload length, it sends the response frame, which contains the length of the maximum payload supported by APS layer.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Command Type - ZIGBEEGETMAXAPSPAYLOADLENGTH (0x39)

Interface Type - MANAGEMENT_INTERFACE (0x1)

Expected: This is expected when ZigBeeGetMaxAPSPayloadLength.

Payload Structure:

```
struct {
    uint8_t           aps_payload_len;
} rsi_MaxApsPayloadLenResp;
```

Payload Parameters:

The **status** can have two values

1. **aps_payload_len:** Maximum aps payload length.

5.3.27 ZigBeeGetKeyResp

Description:

Once device receives ZigBeeGetKey/ZigBeeGetKeyTableEntry request, it validates the input parameters and sends the response frame. If command request was ZigBeeGetKey, then based on the type of Key sent response key information for that corresponding KeyType is expected. If command request was ZigBeeGetKeyTableEntry, then based on the Index sent response key information for that corresponding Index is retrived.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This is expected when ZigBeeGetKey/ZigBeeGetKeyTableEntry command is sent to device from host.

Payload Structure:

```
struct {
    uint8_t  status;
    ZigBeeKeyStructBitmask_t bitmask;
    Security_Key_Types keytype;
    uint8_t  key[16];
    uint32_t outgoingFrameCounter;
    uint32_t incomingFrameCounter;
    uint8_t  sequenceNumber;
    uint8_t  apartnerIEEEAddress[8];
} rsi_GetKeyResp;


typedef enum ZigBeeKeyStructBitmask_Tag {
    g_Key_Has_Sequence_Number_c = 0x01,
    g_Key_Has_Outgoing_Frame_Counter_c = 0x02,
    g_Key_Has_Incoming_Frame_Counter_c = 0x04,
    g_Key_Has_Partner_IEEE_Addr_c = 0x08,
    g_Key_Is_Authorized_c = 0x10
} ZigBeeKeyStructBitmask_t


typedef enum Security_Key_Types_Tag {
    g_Trust_Center_Master_Key_c,
    g_Network_Key_c,
    g_Application_Master_Key_c,
    g_Link_Key_c,
    g_Trust_Center_Link_Key_c,
    g_Next_Network_Key_c
} Security_Key_Types
```

Payload Parameters:

The response payload structure information is given below:

1. **status:** For various status refer

2. **bitmask:** This bitmask indicates the presence of information about that particular field present in bitmask. For e.g., if g_Key_Has_Sequence_Number_c is set then sequence number is present in this payload. The information about each bitfield is provided as:

- **g_Key_Has_Sequence_Number_c:** This indicates that the key has a sequence number associated with Network Key
- **g_Key_Has_Outgoing_Frame_Counter_c:** This indicates that the key has an outgoing frame counter
- **g_Key_Has_Incoming_Frame_Counter_c:** This indicates that the key has an incoming frame counter
- **g_Key_Has_Partner_IEEE_Addr_c**: This indicates that the key has an associated Partner IEEE address and the corresponding value within the ZigBeeKeyStructure_t has been populated with the data

- **g_Key_Is_Authorized_c:** This indicates the key is authorized for use in APS data messages. If the key is not authorized for use in APS data messages it has not yet gone through a key agreement protocol, such as CBKE (i.e. ECC)

3. **keytype**: Type of key sent from host. It is one of key from the defined structure Security_Key_Types.

4. **key**: The actual value of the key to be used for Encryption and Decryption.

5. **outgoingFrameCounter**: This is the outgoing frame counter associated with the key. It will contain valid data based on the ZigBeeKeyStructBitmask_t.

6. **incomingFrameCounter**: This is the incoming frame counter associated with the key. It will contain valid data based on the ZigBeeKeyStructBitmask_t

7. **sequenceNumber**: This is the sequence number associated with the key. It will contain valid data based on the ZigBeeKeyStructBitmask_t

8. **apartnerIEEEAddress**: This is the Partner IEEE Address associated with the key (Link Key). It will contain valid data based on the ZigBeeKeyStructBitmask_t

5.3.28 ZigBeeAddOrUpdateKeyTableEntryResp

Description:

When device receives ZigBeeAddOrUpdateKeyTableEntry command request, it updates or adds the entry in the key table and sends this response frame to host.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This frame is expected when ZigBeeAddOrUpdateKeyTableEntry command frame is sent to device from host.

Payload Structure:

```
struct {
    uint8_t  status;
    uint8_t  Index;
} rsi_StatusResp;
```

Payload Parameters:

1. **Status:** The status would be one of the response value from the table Error Codes.

2. **Index :** Index of the key table where the entry is updated.

5.3.29 ZigBeeFindKeyTableEntryResp

Description:

When device receives ZigBeeFindKeyTableEntry command request, it traverses for the partner's IEEE address in the table entry and returns the key Index if it succeeds in finding IEEE address .

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This frame is expected when ZigBeeAddOrUpdateKeyTableEntry command frame is sent to device from host.

Payload Structure:

```
struct {
    uint8_t  status;
    uint8_t  Index;
} rsi_FindKeyTableResp;
```

Payload Parameters:

1. **Status:** The status would be one of the response value from the table Error Codes.
2. **Index:** Index of the key table where the entry is updated.

5.3.30 ZigBeeGetBindingIndicesResp

Description:

When device receives ZigBeeGetBindingIndices command request, it checks for total number of active indices in binding table and then returns those indices.

Descriptor: The 4-byte and 16-byte descriptors (Rx Operation) are received first, then follows the payload.

Expected: This frame is expected when ZigBeeGetBindingIndices command frame is sent to device from host.

Payload Structure:

```
struct {
    uint8_t  num_of_indices;
    uint8_t  Index[num_of_indices];
} rsi_StatusResp;
```

Payload Parameters:

1. **num_of_indices:** Total number of active indices
2. **Index:** Index numbers of all the active indices

6 ZIGBEE SAPIS

This section contains description about ZigBee API to initialize and configure module in ZigBee mode. this section provides an overview of all the APIs and features present in the stack.

6.1 Management Interface

6.1.1 rsi_zigb_init_stack

Prototype:

```
int16_t  rsi_zigb_init_stack(void);
```

Description:

This API is used to initialize the ZigBee stack.

Parameters:

None

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state.

Returns -4, if packet allocation fails.

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.2 rsi_zigb_reset_stack

Prototype:

```
int16_t  rsi_zigb_reset_stack(void);
```

Description:

This API is used to reset the ZigBee stack.

Parameters:

None

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state.

Returns -4, if packet allocation fails.

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.3 rsi_zigb_set_profile

Prototype:

```
int16_t  rsi_zigb_set_profile(uint8_t profile);
```

Description:

This API is used to set the profile which we are going to use but valid for ZLL profile

Parameters:

Parameters	Data type	Description
Profile	uint8_t	Profile ID for which the stack is going to use 1 – Enable ZLL profile 0 – Disable ZLL profile

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.4 rsi_zigb_update_sas

Prototype:

```
struct {
          uint8_t     a_extended_pan_id[8];
          uint32_t    channel_mask;
          uint8_t     startup_control;
          uint8_t     use_insecure_join;
          uint8_t     scan_attempts;
          uint8_t     parent_retry_threshold;
          uint8_t     a_trust_center_address[8];
          uint8_t     a_network_key[16];
          uint16_t    time_between_scans;
          uint16_t    rejoin_interval;
          uint16_t    max_rejoin_interval;
          uint16_t    indirect_poll_rate;
          uint16_t    a_pan_id;
          uint16_t    network_manager_address;
          uint8_t     a_trustcenter_master_key[16];
          uint8_t     a_preconfigured_link_key[16];
          uint8_t     end_device_bind_timeout;
 }Startup_Attribute_Set_t

int16_t rsi_zigb_update_sas(Startup_Attribute_Set_t *Startup);
```

Description:

This API is used to set the default startup attribute parameters

Parameters:

Parameters	Data type	Description
Startup	Startup_Attribute_Set_t	Pointer to startup attributes structure

Structure Variables:

Parameters	Data type	Valid Range	Description
a_extended_pan_id	uint8_t [8]	0x0000000000000001 – 0xfffffffffffffffe	This field holds the extended PAN ID of the network. In which the device needs to be a member . If the device doesn't know the specific network, then update this 8-

			byte field with zeros otherwise specify the specific 8 bytes extended pan id.
channel_mask	uint32_t	32-bit field	The bits (b11, b12,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan) for each of the 16 valid channels.
startup_control	uint8_t	0x00 – 0x03	This field indicates how the device needs to respond or start depending on the startup control value. **0x00** - Indicates that the device considers itself as a part of the network. indicated by the extended PAN ID attribute. In this case device does not perform any explicit join or rejoin operation. **0x01** - Indicates that the device forms a network with extended PAN ID given by the extended PAN ID attribute. The AIB's attribute APS Designated Coordinator is set to TRUE in this case. **0x02** - Indicates that the device rejoins network with extended PAN ID given by the extended PAN ID attribute. **0x03** - Indicates that the device starts "from scratch"

			and join the network using association. The default value for an un-commissioned device is 0x03.
use_insecure_join	uint8_t	00 = TRUE 01 = FLASE	A flag controlling the use of insecure join at startup.
scan_attempts	uint8_t	1 - 255	Integer value representing the number of scan attempts to make before the NWK layer decides which ZigBee coordinator or router to associate with. This attribute has default value of 5
parent_retry_threshold	uint8_t	3-10	The number of failed attempts to contact a parent that will cause a "find new parent" procedure to be initiated
a_trust_center_address	uint16_t	0x0000-0xFFFF	Address of the network manager.
a_network_key	uint8_t[16]	Variable	The network key.
time_between_scans	uint16_t	1 – 0xFFFF	Time between scans in milliseconds
rejoin_interval	uint16_t	Max value:60	Rejoin interval in seconds
max_rejoin_interval	uint16_t	Max value:3600	Max Rejoin interval in seconds
indirect_poll_rate	uint16_t	In msec	The rate, in milliseconds, to poll the parent
a_pan_id	uint16_t	0x0001 – 0xFFFE	This field indicates the PAN ID of the device.

network_ manager_addre ss	uint16_t	0x0000-0xFFFF	Address of the network manager.
a_trustcenter _ master_key	uint8_t[16]	Variable	The Trust Center master key
a_preconfigur ed_link_key	uint8_t[16]	Variable	The Link key
end_device_bi nd_timeout	uint8_t	1-60	The time the coordinator will wait (in seconds) for a second end device bind request to arrive. The default value is 10.

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.5 rsi_zigb_update_zdo_configuration

Prototype:

```
struct {
      uint8_t     config_permit_join_duration;
      uint8_t     config_NWK_secure_all_frames;
      uint8_t     config_formation_attempts;
      uint8_t     config_scan_duration;
      uint8_t     config_join_attempts;
      uint8_t     config_preconfigured_key;
      uint16_t    a_config_trust_center_short_address;
      uint8_t     automatic_poll_allowed;
      uint8_t     config_authentication_poll_rate;
      uint16_t    config_switch_key_time;
      uint8_t     config_security_level;
      uint8_t     config_aps_ack_poll_time_out;
      uint8_t     a_manufacturer_code[2];
  }ZDO_Configuration_Table_t;
```

```
int16_t
rsi_zigb_update_zdo_configuration(ZDO_Configuration_Table_t
*pzdo_cnf);
```

Description:

This API is used to set the ZDO configuration.

Parameters:

Parameters	Data type	Description
*pzdo_cnf	ZDO_Configuration_Table_t	Pointer to startup ZDO configuration structure

Structure Variables:

Name	Type	Valid Range	Description
config_permit_join_duration	uint8_t	00-0xFF 0x00 Indicates that no devices can join 0xFF Indicates that devices are always allowed to join 0x01 - 0xFE Indicates the time in seconds for which the device allows other devices to join	defines the time for which a coordinator or router device allows other devices to join to itself.
config_NWK_secure_all_f	uint8_t	0-Enable 1-Disabled	defines whether security is

rames			applied for incoming and outgoing network data frames or not
config_formation_attempts	uint8_t	1	the number of times the devices attempts for formation failure.
config_scan_duration	uint8_t	00-0xFE	The field indicates the duration of active scan while performing startup, join or rejoin the network.
config_join_attempts	uint8_t	Default = 02	This field indicates the number of times join is retried once the join fails
config_preconfigured_key	uint8_t	Set to 0x01 if supporting only preconfigured nwk key, or else to be set with 0x02 if requires high security.	This field indicates whether a preconfigured key is already available in the device or not
a_config_trust_center_short_address	I uint16_t	Default 0x0000	This field holds the short address of the TC
automatic_poll_allowed	uint8_t	Enable-0x01 Disable-	This field indicates whether an end device

		0x00(default)	does an auto poll or not.
config_authentication_poll_rate	uint8_t	Default 0x64(100 msec)	The poll rate of end device while waiting for authentication.
config_switch_key_time	uint16_t	Default 0x06	The time after which active key sequence number is changed, once the device receives Switch Key request
config_security_level	uint8_t	0x05	The security level for outgoing and incoming network frames.
config_aps_ack_poll_time_out	uint8_t	0xFA(250 msec)	The maximum number of seconds to wait for an acknowledgement to a transmitted frame.
a_manufacturer_code	uint8_t[2]	0x0000 – 0xFFFF	Manufacturer code

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.6 rsi_zigb_form_network

Prototype:

```
int16_t rsi_zigb_form_network ( uint8_t  RadioChannel,
                  uint8_t power ,uint8_t * pExtendedPanId );
```

Description:

This API allows the Application to establish the Network in the provided channel with the specified Extended PAN ID.

Parameters:

Parameters	Data type	Description
RadioChannel	uint8_t	Channel on which the network needs to be formed. Valid Channels are between 11 – 26 included
power	uint8_t	TX power to be used by the device. Range of TX power is about 0 – 12dBm.
pExtendedPanId	uint8_t	Pointer to extended PANID array of 8 bytes.

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.7 rsi_zigb_join_network

Prototype:

```
int16_t rsi_zigb_join_network(uint8_t DeviceType,
                  uint8_t RadioChannel,
                  uint8_t power ,
                  uint8_t *pExtendedPanId);
```

Description:

This API allows the Application to join the Network in the provided channel with the (coordinator of) specified Extended PAN Id.

Parameters:

Parameters	Data type	Description
DeviceType	uint8_t	0x01 – Router 0x02 – End-Device
RadioChannel	uint8_t	Channel on which the network needs to be formed. Valid Channels are between 11 – 26 included
power	uint8_t	TX power to be used by the device. Range of TX power is about 0 – 12dBm.
pExtendedPanId	uint8_t	Pointer to extended PANID array of 8 bytes.

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.8 rsi_zigb_permit_join

Prototype:

```
int16_t rsi_zigb_permit_join(uint8_t PermitDuration);
```

Description:

This API allows the Application to enable join permit on the device for the specified duration in seconds.

Parameters:

Parameters	Data type	Description
PermitDuration	uint8_t	The length of time in seconds during which the ZigBee coordinator or router will allow associations. The valid values are as below 0x00 = Disabled. 0xFF = Always allowed associations. 0x01 – 0xFE = Associations allowed for this timeout

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.9 rsi_zigb_leave_network

Prototype:

```
int16_t rsi_zigb_leave_network(void);
```

Description:

This API allows to perform self leave from the network.

Parameters:

None

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.10 rsi_zigb_initiate_scan

Prototype:

```
int16_t rsi_zigb_initiate_scan(uint8_t scanType,
                  uint32_t channelMask,
                  uint8_t duration);
```

Description:

This API allows the Application to initiate Scan of specified type in the specified channel mask for the specified duration.

Parameters:

Parameters	Data type	Description
scanType	uint8_t	0x00 – Energy Detection scan 0x01 – Active scan
ChannelMask	uint32_t	The five most significant bits (b27,..., b31) and 11 least significant bits (b0,b1,…b10) are reserved. The middle 16 bits (b11, b12,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan).
Duration	uint8_t	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is (aBaseSuperframeDuration * (2^n + 1)) symbols, where n is the value of the ScanDuration parameter .

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.11 rsi_zigb_stop_scan

Prototype:

```
int16_t rsi_zigb_stop_scan(void);
```

Description:

This API allows the Application to stop the scan that was initiated.

Parameters:

None

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.12 rsi_zigb_network_state

Prototype:

```
int16_t rsi_zigb_network_state(void);
```

Description:

This API allows the Application to know if the device is in the process of joining, already Joined or leaving the network.

Parameters:

None

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.13 rsi_zigb_stack_is_up

Prototype:

```
int16_t rsi_zigb_stack_is_up(void);
```

Description:

This API is used to know whether the stack is running or not. It returns success after joining to coordinator for End-Device and Router. For coordinator it returns success after forming the network.

Parameters:

None

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.14 rsi_zigb_get_self_ieee_address

Prototype:

```
int16_t rsi_zigb_get_self_ieee_address(uint8_t* ieee_addr);
```

Description:

This API allows to application to read the device self IEEE extended address.

Parameters:

Parameters	Data type	Description
ieee_addr	uint8_t*	Pointer to IEEE address (array of 8 bytes) in which the device self IEEE address to be copied.

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is other than above, please refer ZigBee error code table for description.

6.1.15 rsi_zigb_is_it_self_ieee_address

Prototype:

```
int16_t rsi_zigb_is_it_self_ieee_address(uint8_t *pIEEEAddress);
```

Description:

This API allows the application to know the given Extended address is self IEEE address.

Parameters:

Parameters	Data type	Description
pIEEEAddress	uint8_t*	Pointer to IEEE address (array of 8 bytes) which has to verified.

Return Value:

On Success : g_TRUE_c, if IEEE address is the local node's ID.

On Failure : g_FALSE_c

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.16 rsi_zigb_get_self_short_address

Prototype:

```
int16_t rsi_zigb_get_self_short_address(void);
```

Description:

This api allows the application know self short address.

Parameters:

None

Return Value:

On Success: 16-bit short self address.

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is other than above, please refer ZigBee error code table for description.

6.1.17 rsi_zigb_set_manufacturer_code_for_node_desc

Prototype:

```
int16_t rsi_zigb_set_manufacturer_code_for_node_desc(uint16_t
code);
```

Description:

This api allows the user to ser manufacturer code in the node descriptor.

Parameters:

Parameters	Data type	Description
code	uint16_t	The 16-bit manufacturer code for the local node. Range:0x0000 -0xFFFF.

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.18 rsi_zigb_set_power_descriptor

Prototype:

```
struct  {
        uint8_t         current_powermode_avail_power_sources;
        uint8_t
current_powersource_currentpowersourcelevel;
      }Node_Power_Descriptor_t;


int16_t rsi_zigb_set_power_descriptor(Node_Power_Descriptor_t *);
```

Description:

This api allows the application to set power descriptor for the device.

Structure Variables:

Name	Type	Valid Range	Description
current_powermode_avail_power_sources	uint8_t	00-0xFF	the first 4 bits of LSB gives the current sleep/ power saving mode of

			the node and MSB 4 bits gives the power sources available in this node.
current_powersource_currentpowersourcelevel	uint8_t	00-0xFF	the first 4 bit of LSB gives the current power source and 4 bits of MSB gives the current power source level.

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.19 rsi_zigb_set_maxm_incoming_txfr_size

Prototype:

```
int16_t rsi_zigb_set_maxm_incoming_txfr_size(uint16_t size);
```

Description:

The api allows the application to specify the maximum incoming transfer size the device is capable of.

Parameters:

Parameters	Data type	Description
size	uint16_t	The maximum incoming transfer size for the local node. Range:0-128

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.20 rsi_zigb_set_maxm_outgoing_txfr_size

Prototype:

```
int16_t rsi_zigb_set_maxm_outgoing_txfr_size(uint16_t);
```

Description:

The api allows the application to specify the maximum outgoing transfer size the device is capable of.

Parameters:

Parameters	Data type	Description
Size	uint16_t	The maximum outgoing transfer size for the local node. Range:0-128

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.21 rsi_zigb_set_operating_channel

Prototype:

```
int16_t rsi_zigb_set_operating_channel(uint8_t channel);
```

Description:

The api allows the application to set the operating channel.

Parameters:

Parameters	Data type	Description
Channel	uint8_t	The desired radio channel. Range:11 to 26.

Return Value:

On Success : 0

On Failure : non-zero

Returns a non-zero value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.22 rsi_zigb_get_device_type

Prototype:

```
int16_t rsi_zigb_get_device_type(uint8_t dev_type);
```

Description:

The api allows the application to get the device type.

Parameters:

Parameters	Data type	Description
dev_type	uint8_t	The type of the device. 0 – Coordinator. 1 – Router.

		2 – EndDevice.

Return Value:

On Success : 0,the dev_type parameter is updated.

On Failure : non-zero

Returns a non-zero value for unknown device or if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.23 rsi_zigb_get_operating_channel

Prototype:

```
int16_t rsi_zigb_get_operating_channel(void);
```

Description:

The api allows the application to get the operating channel.

Parameters:

None.

Return Value:

On Success : The channel number.

On Failure : other than 11-26.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.24 rsi_zigb_get_short_pan_id

Prototype:

```
int16_t rsi_zigb_get_short_pan_id(void);
```

Description:

The api allows the application to get the short panid.

Parameters:
None.

Return Value:

On Success : 16 bit Pan Id.

On Failure : 0XFFFF.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.25 rsi_zigb_get_extended_pan_id

Prototype:

```
int16_t rsi_zigb_get_extended_pan_id(uint8_t *p_extended_panid );
```

Description:

The api allows the application to get the extended pan id.

Parameters:

Parameters	Data type	Description
p_extended_panid	uint8_t*	Pointer to the array in which the extended pan id is to be updated.

Return Value:

On Success : 0, the extended pan-id is updated in p_extended_panid.

On Failure : non-zero

Returns a negative value if command is issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.26 rsi_zigb_get_endpoint_id

Prototype:

```
int16_t rsi_zigb_get_endpoint_id(uint8_t index);
```

Description:

The api allows the application to get the endpoint id.

Parameters:

Parameters	Data type	Description
index	uint8_t	Indicates the index of the array. This value should be less than the Number of endpoints.

Return Value:

On Success : The valid Endpoint ID located in the specified index.

On Failure : g_INVALID_ENDPOINT_ID_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.27 rsi_zigb_get_simple_descriptor

Prototype:

```
typedef uint16_t profile_id_t,cluster_id_t;

typedef struct Simple_Descriptor_Tag {
        profile_id_t     app_profile_id;
        uint16_t         app_device_id;
        uint8_t          app_device_version;
        uint8_t          incluster_count;
        cluster_id_t const *p_incluster_list;
        uint8_t          outcluster_count;
        cluster_id_t const *p_outcluster_list;
      }Simple_Descriptor_t;
```

```
int16_t rsi_zigb_get_simple_descriptor(
                  uint8_t endpointId,
                  Simple_Descriptor_t *p_simple_desc);
```

Description:

The api allows the application to get the simple descriptor.

Parameters:

Parameters	Data type	Description
endpoint_id	uint8_t	The Endpoint on which these clusters are defined
p_simple_desc	Simple_Descriptor_t *	Pointer to simple descriptor of specified endpoint.

Name	type	Range	Description
app_profile_id	profile_id_t	0x0000 – 0xffff	The Endpoint on which these clusters are defined
app_device_id	uint16_t	0x0000 - 0xfff7	The address of the designated network channel manager function
app_device_version	uint8_t	1-254	The version of the ZigBee protocol in use in the discovered network.
incluster_count	uint8_t	0x00 – 0x0f	The number of Input Clusters
p_incluster_list	cluster_id_t	-	pointer to buffer holding input clusters.
outcluster_count	uint8_t	0x00 – 0x0f	The number of Output Clusters
p_outcluster_list	cluster_id_t	-	pointer to buffer holding output clusters.

Structure: Simple_Descriptor_t

Table : Simple Descriptor structure

Return Value:

On Success : g_TRUE_c.

On Failure : g_FALSE_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.28 rsi_zigb_set_simple_descriptor

Prototype:

```
int16_t rsi_zigb_set_simple_descriptor(
                  uint8_t endpointId,
                  Simple_Descriptor_t *simple_desc);
```

Description:

The api allows the application to set the simple descriptor.

Parameters:

Parameters	Data type	Description
endpoint_id	uint8_t	The Endpoint on which the clusters are defined.
p_simple_desc	Simple_Descriptor_t *	Pointer to simple descriptor of specified endpoint.

Return Value:

On Success : g_TRUE_c.

On Failure : g_FALSE_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.29 rsi_zigb_get_endpoint_cluster

Prototype:

```
int16_t rsi_zigb_get_endpoint_cluster(uint8_t EndPointId, uint8_t
ClusterType,uint8_t ClusterIndex)
```

Description:

The api allows the application to read the endpoint's cluster in the specified list at the specified end-point index.

Parameters:

Parameters	Data type	Description
EndPointId	uint8_t	The 8-bit endpoint id whose cluster id needs to be retrieved
ClusterType	uint8_t	Indicates if the incluster list should be read or outcluster list to be read. 0 indicates incluster list and 1 indicates outcluster list.
ClusterIndex	uint8_t	Indicates the index of the list of which cluster id is to be read. This index should be less than the number of clusters supported in the list as read in the simple descriptor.

Return Value:

On Success : Cluster id of the endpoint's simple descriptor located at the specified index.

On Failure : g_INVALID_CLUSTER_ID_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.30 rsi_zigb_get_short_addr_for_specified_ieee_addr

Prototype:

```
int16_t rsi_zigb_get_short_addr_for_specified_ieee_addr(uint8_t *
pIEEEAddress);
```

Description:

The api allows the application to get the 16-bit short address of the device for the given 64-bit IEEE address.

Parameters:

Parameters	Data type	Description
pIEEEAddress	uint8_t*	Pointer to IEEE address whose 16-bit short address is to be determined

Return Value:

On Success : 16-bit short address of the corresponding 64-bit IEEE address if the address is known.

On Failure : INVALID_SHORT_ADDRESS

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.31 rsi_zigb_get_ieee_addr_for_specified_short_addr

Prototype:

```
int16_t rsi_zigb_get_ieee_addr_for_specified_short_addr(
                  uint16_t shortAddr,
                  uint8_t* ieee_addr);
```

Description:

The api allows the application to get the 64-bit IEEE address of the device for the given 16-bit Short address.

Parameters:

Parameters	Data type	Description
shortAddr	uint16_t	shortAddr gives the 16-bit short address of which the corresponding 64-bit IEEE

		address need to be determined
ieee_addr	uint8_t*	Pointer to location where the IEEE address needs to be copied.

Return Value:

On Success : **g_TRUE_c,** if successfully retrieved IEEE address from neighbor table or Address map table.

On Failure : **g_FALSE_c**

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.32 rsi_zigb_read_neighbor_table_entry

Prototype:

```
typedef struct ZigBeeNeighborTableEntry_Tag {
                  uint16_t          shortId;
                  uint8_t           averageLqi;
                  uint8_t           incomingCost;
                  uint8_t           outgoingCost;
                  uint8_t           age;
                  uint8_t           aIEEEAddress[8];
                  }ZigBeeNeighborTableEntry_t;


int16_t rsi_zigb_read_neighbor_table_entry(
                  uint8_t Index,
                  ZigBeeNeighborTableEntry_t *neigbor_table);
```

Description:

The api allows the application to read the Neighbor table entry in the specified index.

Parameters:

Parameters	Data type	Description
Index	uint8_t	Indicates index from where the neighbor table entry is to be retrieved.
neigbor_table	ZigBeeNeighbor	Pointer to location where the NeighbortableEntry needs to

	TableEntry_t*	be copied.

Structure: ZigBeeNeighborTableEntry_t

Name	type	Range	Description
shortId	uint16_t	0x0000 – 0xffff	The neighbor's two byte short address
averageLqi	uint8_t	0x00-0xf0	An exponentially weighted moving average of the link quality values of incoming packets from this neighbor as reported by the PHY.
incomingCost	uint8_t	1-7	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 .
outgoingCost	uint8_t	1-7	The outgoing cost for this neighbor, obtained from the most recently received neighbor exchange message from the neighbor
age	uint8_t	3-16	The number of aging periods elapsed since a neighbor exchange message was last received from this neighbor. An entry with an age greater than 3 is considered stale and may be reclaimed. The aging period is 16 seconds
aIEEEAddress	uint8_t[8]	-	The 8 byte IEEE address of the neighbor

Return Value:

On Success : **0**.

On Failure : **ZigBee_Invalid_Argument**

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.33 rsi_zigb_get_route_table_entry

Prototype:

```
typedef struct ZigBeeRoutingTableEntry_Tag {
  uint16_t         destAddr;
  uint16_t         nextHop;
  uint8_t          status;
  uint8_t          age;
  uint8_t          concentratorType;
  uint8_t          routeRecordState;
}ZigBeeRoutingTableEntry_t;

int16_t rsi_zigb_get_route_table_entry(
                  uint8_t Index,
                  ZigBeeRoutingTableEntry_t *routing_table);
```

Description:

The api allows the application to read the Routing table entry in the specified index.

Parameters:

Parameters	Data type	Description
Index	uint8_t	Indicates index from where the neighbor table entry is to be retrieved.
routing_table	ZigBeeRoutingTableEntry_t *	Pointer to location where the Route table Entry needs to be copied.

Structure: ZigBeeNeighborTableEntry_t

Name	type	Range	Description
destAddr	uint16_t	0x0000 – 0xffff	short id of the destination
nextHop	uint16_t	0x0000 – 0xffff.	short address of the next hop to this destination
status	uint8_t	1-7	Indicates whether this entry is active (0), being discovered (1), or unused (0x3).
age	uint8_t	1-7	The number of seconds since this route entry was last used to send a packet
concentratorType	uint8_t	0-2	Indicates whether this destination is a High RAM Concentrator (2), a Low RAM Concentrator (1), or not a concentrator (0).

routeRecordState	uint8_t	0-2	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

Return Value:

On Success : **0**.

On Failure :

- ZigBee Index Out Of Range : Accessing entry is out of range in the table.
- ZigBee Invalid Argument : Argument passed for API is invalid .

  Returns a negative value if command issued in wrong state and packet allocation failure.

  Returns -3, if command issued in wrong state

  Returns -4, if packet allocation fails

  If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.34 rsi_zigb_get_neighbor_table_entry_count

Prototype:

```
int16_t rsi_zigb_get_neighbor_table_entry_count(void);
```

Description:

The api allows the application to know the count of active neighbor table entries.

Parameters:

None.

Return Value:

On Success : Total count of active neighbor table entries in the neighbor table.

On Failure : non-zero

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.35 rsi_zigb_get_child_short_address_for_the_index

Prototype:

```
int16_t rsi_zigb_get_child_short_address_for_the_index(uint8_t
ChildIndex);
```

Description:

The api allows the application to read the 16-bit short address of the child in the specified index.

Parameters:

Parameters	Data type	Description
ChildIndex	uint16_t	Indicates the index from where the 16-bit short address needs to be retrieved

Return Value:

On Success **:** The child address.

On Failure : g_INVALID_ADDRESS_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.36 rsi_zigb_get_child_index_for_specified_short_addr

Prototype:

```
int16_t
rsi_zigb_get_child_index_for_specified_short_addr(uint16_t
childShortAddr)
```

Description:

The api allows the application to get the index for the specified 16-bit child address.

Parameters:

Parameters	Data type	Description
childShortAddr	uint16_t	The 16-bit short address whose index need to be determined

Return Value:
On Success **:** index of the child address received in the input parameter.

On Failure : m_NO_ENTRY_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.37 rsi_zigb_get_child_details

Prototype:

```
int16_t rsi_zigb_get_child_details(
                        uint8_t Index,
                        uint8_t *ieee_addr,
                        uint8_t DeviceType);
```

Description:

The api allows the application to get the child details at the specified child index.

Parameters:

Parameters	Data type	Description
Index	uint8_t	The index of the child of interest.
ieee_addr	uint8_t*	The child's EUI64 is copied into here.
DeviceType	uint8_t	The child's node type is copied into here. 0 – Coordinator. 1 – Router. 2 – EndDevice.

Return Value:
On Success **: 0,**

On Failure : ZigBeeUnknownDevice

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.38 rsi_zigb_end_device_poll_for_data

Prototype:

```
int16_t rsi_zigb_end_device_poll_for_data( void );
```

Description:

The api allows the application to poll the parent for data.

Parameters:

None.

Return Value:

On Success **: 0,**

On Failure : ZigBee_Invalid_Call

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.39 rsi_zigb_read_count_of_child_devices

Prototype:

```
int16_t rsi_zigb_read_count_of_child_devices(void);
```

Description:

The api allows the application to read the number of child devices on the node.

Parameters:

None.

Return Value:
On Success **:** Number of children joined.

On Failure : negative value

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.40 rsi_zigb_read_count_of_router_child_devices

Prototype:

```
int16_t rsi_zigb_read_count_of_router_child_devices(void);
```

Description:

The api allows the application to read the number of child devices on the node.

Parameters:

None.

Return Value:
On Success **:** Number of router children joined.

On Failure : negative value

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.41 rsi_zigb_get_parent_short_address

Prototype:

```
int16_t rsi_zigb_get_parent_short_address(void);
```

Description:

The api allows the application to get the parent's 16 bit short address.

Parameters:

None.

Return Value:

On Success **:** parent short address.

On Failure : g_INVALID_ADDRESS_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.42 rsi_zigb_get_parent_ieee_address

Prototype:

```
int16_t rsi_zigb_get_parent_ieee_address(uint8_t *ieee_addr);
```

Description:

The api allows the application to read it parent's 64-bit IEEE address.

Parameters:

Parameters	Data type	Description
ieee_addr	uint8_t*	Pointer to location where parent's 64-bit IEEE address should be copied

Return Value:

On Success **: 0,**

On Failure : negative value

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.43 rsi_zigb_initiate_energy_scan_request

Prototype:

```
int16_t rsi_zigb_initiate_energy_scan_request(uint16_t
DestAddr,uint32_t ScanChannels, uint8_t ScanDuration, uint16_t
ScanRetry);
```

Description:

The api allows the application to request energy scan be performed and its results returned. This request may only be sent by the current network manager and must be unicast, not broadcast.

Parameters:

Parameters	Data type	Description
DestAddr	uint16_t	Indicates the network address of the device to perform the scan. Range:0x0000-0xFFFF
ScanChannels	uint32_t	The five most significant bits (b27,..., b31) and 11 least significant bits (b0,b1,…b10) are reserved. The middle 16 bits (b11, b12,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan).
ScanDuration	uint8_t	Indicates How long to scan on each channel. Allowed values are 0 – 5.
ScanRetr	uint16_t	Indicates The number of scans to be performed on each channel (1-8)

Return Value:

On Success **: 0,**

On Failure : non-zero

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.44 rsi_zigb_broadcast_nwk_manager_request

Prototype:

```
int16_t rsi_zigb_broadcast_nwk_manager_request(uint16_t
NWKManagerShortAddr, uint32_t ActiveChannels);
```

Description:

The api allows the application to broadcasts a request to change the channel. This request may only be sent by the current Network manager.

Parameters:

Parameters	Data type	Description
NWKManagerShort Addr	uint16_t	Indicates the 16-bit network address of the Network Manager.
ActiveChannels	uint32_t	Indicates the new active channel mask. The five most significant bits (b27,..., b31) and 11 least significant bits (b0,b1,…b10) are reserved. The middle 16 bits (b11, b12,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan).

Return Value:

On Success **: 0,**

On Failure : non-zero

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.45 rsi_zigb_zdp_send_nwk_addr_request

Prototype:

```
int16_t rsi_zigb_zdp_send_nwk_addr_request(uint8_t *
pIEEEAddrOfInterest, BOOL RequestType, uint8_t StartIndex);
```

Description:

The api allows the application to send ZDP network address request to determine the 16-bit short address of the device whose IEEE address is known.

Parameters:

Parameters	Data type	Description
pIEEEAddrOfInterest	uint8_t*	Pointer to location of IEEE address whose 16-bit Network address is to be determined
RequestType	BOOL	boolean if TRUE indicates single device response if FALSE indicates extended device response.
StartIndex	uint8_t	Start index of the child devices list.

Return Value:

On Success **: 0,**

On Failure : g_FAILURE_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.46 rsi_zigb_zdp_send_ieee_addr_request

Prototype:

```
int16_t rsi_zigb_zdp_send_ieee_addr_request(uint16_t
shortAddress, BOOL RequestType,uint8_t StartIndex, BOOL
APSAckRequired)
```

Description:

The api allows the application to send ZDP IEEE address request to determine the 16-bit short address of the device whose IEEE address is known.

Parameters:

Parameters	Data type	Description
shortAddress	uint16_t	Pointer to location of short address whose IEEE address is to be determined.
RequestType	BOOL	TRUE indicates single device response if FALSE indicates extended device response.
StartIndex	uint8_t	The index of the first child to list in the response. Ignored if the RequestType is single device.
APSAckRequired	BOOL	TRUE indicates APS ack is required

Return Value:

On Success **: 0,**

On Failure : g_FAILURE_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.47 rsi_zigb_zdp_send_device_announcement

Prototype:

```
int16_t rsi_zigb_zdp_send_device_announcement(void);
```

Description:

The api allows the application to send a broadcast for a ZDO Device announcement. Normally, it is NOT required to call this as the stack automatically sends a device announcement during joining or rejoining, as per the spec. However, if the device wishes to broadcast device announcement it can do through this call.

Parameters:

None.

Return Value:

On Success **: 0,**

On Failure : ZigBee_Device_Down.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.48 rsi_zigb_send_match_descriptors_request

Prototype:

```
int16_t rsi_zigb_send_match_descriptors_request(
            uint16_t shortAddress,
            uint16_t ProfileId,
            uint8_t *InClusterList,
            uint8_t InClusterCnt,
            uint8_t *OutClusterList,
            uint8_t OutClusterCnt,
            BOOL APSAckRequired,
            uint16_t dstAddress);
```

Description:

The api allows the application to send a match descriptor request to a destination device.

Parameters:

Parameters	Data type	Description
shortAddress	uint16_t	The device whose matching endpoints are desired. The request can be sent unicast or broadcast ONLY to the "RX-on-when-idle- address" (0xFFFD) If sent as a broadcast, any node

		that has matching endpoints will send a response.
ProfileId	uint16_t	The application profileId to match
InClusterList	uint8_t	The list of input clusters.
InClusterCnt	uint8_t	Number of input clusters.
OutClusterList	uint8_t	The list of output clusters.
OutClusterCnt	uint8_t	Number of output clusters.
APSAckRequired	BOOL	TRUE indicates APS ack is required
dstAddress	uint16_t	Destination short address.

Return Value:
On Success **: 0,**
On Failure : ZigBee_Failure

.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.48.1 rsi_zigb_active_endpoints_request

Prototype:

```
int16_t rsi_zigb_active_endpoints_request(uint16_t shortAddress,
uint8_t APSAckRequired)
```

Description:

The api allows the application to send ZDP Active Endpoint request.

Parameters:

Parameters	Data type	Description

shortAddress	uint16_t	Device short address whose active endpoints needs to be obtained.
APSAckRequired	BOOL	TRUE indicates APS ack is required

Return Value:

On Success **: 0,**

On Failure : g_FAILURE_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.49 rsi_zigb_zdp_send_power_descriptor_request

Prototype:

```
int16_t rsi_zigb_zdp_send_power_descriptor_request(uint16_t
shortAddress, uint8_t APSAckRequired)
```

Description:

The api allows the application to power descriptor request.

Parameters:

Parameters	Data type	Description
shortAddress	uint16_t	Device short address whose power descriptor needs to be obtained.
APSAckRequired	BOOL	TRUE indicates APS ack is required

Return Value:

On Success **: 0,**

On Failure : g_FAILURE_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.50 rsi_zigb_zdp_send_node_descriptor_request

Prototype:

```
int16_t rsi_zigb_zdp_send_node_descriptor_request(uint16_t shortAddress, uint8_t APSAckRequired);
```

Description:

The api allows the application to node descriptor request.

Parameters:

Parameters	Data type	Description
shortAddress	uint16_t	Device short address whose node descriptor needs to be obtained.
APSAckRequired	BOOL	TRUE indicates APS ack is required

Return Value:

On Success **: 0,**

On Failure : g_FAILURE_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.51 rsi_zigb_simple_descriptor_request

Prototype:

```
int16_t rsi_zigb_simple_descriptor_request(uint16_t shortAddress, uint8_t EndPointId);
```

Description:

The api allows the application to request for the simple descriptor for a target device.

Parameters:

Parameters	Data type	Description
shortAddress	uint16_t	Device short address whose simple descriptor needs to be obtained. Range:0x0000 – 0xFFFF.
APSAckRequired	BOOL	TRUE indicates APS ack is required

Return Value:

On Success **: 0,**

On Failure : g_FAILURE_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.1.52 rsi_zigb_get_address_map_table_entry

Prototype:

```
typedef struct APSME_Address_Map_Table_Tag {
              uint8_t          a_IEEE_addr[8];
              uint16_t         nwk_addr;
            }APSME_Address_Map_Table_t;


APSME_Address_Map_Table_t*
rsi_zigb_get_address_map_table_entry(uint8_t Index);
```

Description:

The api allows the application to get the address map table entry for the specified index.

Parameters:

Parameters	Data type	Description
Index	uint8_t	Specifies which entry in the Address Map table

Structure: `APSME_Address_Map_Table_t`

Name	type	Range	Description
a_IEEE_addr	uint8_t[8]	-	indicates extended 64-bit IEEE address.
nwk_addr	uint16_t	0x0000 – 0xffff.	16 bit network address.

Return Value:

On Success **:** Address map table is updated.

On Failure : The 64 bit field is updated with all 0xFFs.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.2 Data Interface

6.2.1 rsi_zigb_send_unicast_data

Prototype:

```
typedef enum
{
  ZigBee_Outgoing_Direct,
  ZigBee_Via_Address_Map,
  ZigBee_Via_Binding_Table,
  ZigBee_Via_Multicast,
  ZigBee_Broadcast
}ZigBee_Outgoing_Msg_Type;


typedef struct {
  uint8_t          DestEndpoint;
  uint8_t          SrcEndpoint;
  ProfileID        ProfileId;
  ClusterID        ClusterId;
  uint8_t          AsduLength;
  uint8_t          TxOptions;
  uint8_t          Radius;
  uint8_t          aReserved[0x31];
  uint8_t          aPayload[0x33];
}ZigBeeAPSDEDataRequest_t;

typedef union Address_Tag {
```

```
  uint16_t short_address;
  uint8_t IEEE_address[8];
} Address;


int16_t rsi_zigb_send_unicast_data(
            ZigBee_Outgoing_Msg_Type msgType,
            Address DestAddress,
            ZigBeeAPSDEDataRequest_t *pAPSDERequest);
```

Description:

The api allows the application to to initiate APSDE data request to the specified destination address.

Parameters:

Parameters	Data type	Description
msgType	uint8_t	Type of transmission taking place.
DestAddress	Address	Address of the destination device
pAPSDERequest	ZigBeeAPSDEDataRequest_t *	Pointer to memory where data request frame is stored.

Name	type	Range	Description
short_address	uint16_t	0x00 – 0xff	16-bit short address.
IEEE_address	uint8_t[8]	-	64-bit IEEE extended address.

Structure: Address

Structure: ZigBeeAPSDEDataRequest_t

Name	type	Range	Description
DestEndpoint	uint8_t	0x00 – 0xff	This parameter shall be present if, and onlyif, the DstAddrMode parameter has a value of 0x02 or 0x03 and, if present, shall be either the number of the individual endpoint of the entity to which the ASDU is being transferred or the broadcast endpoint (0xff).

SrcEndpoint	uint8_t	0x00 – 0xfe	The individual endpoint of the entity from which the ASDU is being transferred.
ProfileId	uint16_t	0x0000 – 0xffff	The identifier of the profile for which this frame is intended.
ClusterId	uint16_t	0x0000 – 0xffff	The identifier of the object for which this frame is intended
AsduLength	uint8_t	0x00 - 256*(NsduLength - apscMinHeader Overhead)	The number of octets comprising the ASDU to be transferred. The maximum length of an individual APS frame payload is given as NsduLength - *apscMinHeaderOverhead*. Assuming fragmentation is used, there can be 256 such blocks comprising a single maximum sized ASDU.
TxOptions	uint8_t	0000 0000 – 00011111	The transmission options for the ASDU to be transferred. These are a bitwise OR of one or more of the following: 0x01 = Security enabled transmission 0x02 = Use NWK key 0x04 = Acknowledged transmission 0x08 = Fragmentation permitted 0x10 = Include extended nonce in APSsecurity frame
Radius	uint8_t	0x00-0xff	The distance, in hops, that a transmitted frame will be allowed to travel through the network.
aReserved	uint8_t[0x31]	-	Reserved bytes for payload.
aPayload	uint8_t[0x33]	-	Payload.

Return Value:

On Success **:** ZigBee_Success.

On Failure :

ZigBee_Invalid_Argument/ ZigBee_No_Buffer.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.2.2 rsi_zigb_send_group_data

Prototype:

```
typedef uint16_t GroupID;
int16_t rsi_zigb_send_group_data( GroupID GroupAddress,
ZigBeeAPSDEDataRequest_t * pAPSDERequest);
```

Description:

The api allows the application to initiate APSDE data request to the specified Group address.

Parameters:

Parameters	Data type	Description
GroupAddress	GroupID	Indicates the group id to which the data is transmitted.
pAPSDERequest	ZigBeeAPSDEDataRequest_t *(refer rsi_zigb_send_unicast_data for the structure definition)	Pointer to memory where data request frame is stored.

Return Value:

On Success **:** ZigBee_Success.

On Failure :

ZigBee_Invalid_Argument/ ZigBee_No_Buffer.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.2.3 rsi_zigb_send_broadcast_data

Prototype:

```
int16_t rsi_zigb_send_broadcast_data(
                ZigBeeAPSDEDataRequest_t * pAPSDERequest);
```

Description:

The api allows the application to broadcast APSDE data request.

Parameters:

Parameters	Data type	Description
pAPSDERequest	ZigBeeAPSDEDataRequest_t * (refer rsi_zigb_send_unicast_data for the structure definition)	Pointer to memory where data request frame is stored.

Return Value:
On Success **:** ZigBee_Success.
On Failure :

ZigBee_Invalid_Argument/ ZigBee_No_Buffer.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.2.4 rsi_zigb_get_max_aps_payload_length

Prototype:

```
int16_t rsi_zigb_get_max_aps_payload_length(void);
```

Description:

The api allows the application to get the maximum size of the payload that the Application Support sub-layer will accept. The size depends on the security level in use. The value is the same as that found in the node descriptor.

Parameters:

None.

Return Value:
On Success **:** maximum APS payload length.

On Failure : negative value.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.3 Security Interface

KeyType	Value
g_Trust_Center_Master_Key_c (Reserved)	**0x0**
g_Network_Key_c	**0x1**
g_Application_Master_Key_c (Reserved)	**0x2**
g_Link_Key_c (Reserved)	**0x3**
g_Trust_Center_Link_Key_c	**0x4**

Table : Key Types

6.3.1 rsi_zigb_get_key

Prototype:

```
typedef enum Security_Key_Types_Tag {
                  g_Trust_Center_Master_Key_c,
                  g_Network_Key_c,
                  g_Application_Master_Key_c,
                  g_Link_Key_c,
                  g_Trust_Center_Link_Key_c,
                  g_Next_Network_Key_c
                  } Security_Key_Types;


int16_t rsi_zigb_get_key(Security_Key_Types keytype);
```

Description:

The api allows the application to gets the specified key and its associated data. This can retrieve the Link Key, Current Network Key, or Next Network Key.

.

Parameters:

Parameters	Data type	Description
keytype	Security_Key_ Types	key type.(refer Table 4)

Return Value:

On Success **:** ZigBee_Success.

On Failure :

ZigBee_Invalid_Argument/ g_NO_KEY_c/ ZigBee_Failure

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.3.2 rsi_zigb_have_link_key

Prototype:

```
int16_t rsi_zigb_have_link_key(uint8_t *pRemoteDeviceIEEEAddr);
```

Description:

The api allows the application to check a link key is available for securing messages sent to the remote device.

Parameters:

Parameters	Data type	Description
pRemoteDeviceIE EEAddr	uint8_t*	The long address of some other device in the network.

Return Value:

On Success **:** g_TRUE_c.

On Failure : g_FALSE_c

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.3.3 rsi_zigb_request_link_key

Prototype:

```
int16_t rsi_zigb_request_link_key(uint8_t*
TrustCenterIEEEAddr, uint8_t* PartnerIEEEAddr);
```

Description:

The api allows the application to get the link key for the specified IEEE address.

Parameters:

Parameters	Data type	Description
TrustCenterIEEE Addr	uint8_t*	The IEEE address of the Trust Centre device.
PartnerIEEEAddr	uint8_t*	The IEEE address of the partner device.

Return Value:

On Success **:** ZigBee_Success.

On Failure : ZigBee_Failure/ ZigBee_Invalid_Argument.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.3.4 rsi_zigb_get_key_table_entry

Prototype:

```
typedef enum Security_Key_Types_Tag {
                    g_Trust_Center_Master_Key_c,
                    g_Network_Key_c,
                    g_Application_Master_Key_c,
                    g_Link_Key_c,
                    g_Trust_Center_Link_Key_c,
```

```
                    g_Next_Network_Key_c
                    } Security_Key_Types;

typedef enum ZigBeeKeyStructBitmask_Tag {
                    g_Key_Has_Sequence_Number_c = 0x01,
                    g_Key_Has_Outgoing_Frame_Counter_c = 0x02,
                    g_Key_Has_Incoming_Frame_Counter_c = 0x04,
                    g_Key_Has_Partner_IEEE_Addr_c = 0x08,
                    g_Key_Is_Authorized_c = 0x10
                    } ZigBeeKeyStructBitmask_t;

typedef struct ZigBeeKeyStructure_Tag {
                    ZigBeeKeyStructBitmask_t bitmask;
                    Security_Key_Types       type;
                    uint8_t          key[16];
                    uint32_t         outgoingFrameCounter;
                    uint32_t         incomingFrameCounter;
                    uint8_t          sequenceNumber;
                    uint8_t
              apartnerIEEEAddress[g_EXTENDED_ADDRESS_LENGTH_c];
                    }ZigBeeKeyStructure_t;


int16_t rsi_zigb_get_key_table_entry (uint8_t Index,
ZigBeeKeyStructure_t *keyStruct);
```

Description:

The api allows the application to get the link key for the specified IEEE address.

Parameters:

Parameters	Data type	Description
Index	uint8_t	The index in the key table of the entry to get.
keyStruct	ZigBeeKeyStructure_t *	A pointer to the location of an **ZigBeeKeyStructure_t** that will contain the results retrieved by the stack.

Parameters: ZigBeeKeyStructBitmask_t

Parameters	Description
g_Key_Has_Sequence_Number_c	This indicates that the key has a sequence number associated with Network Key
g_Key_Has_Outgo	• This indicates

ing_Frame_Count er_c	that the key has an outgoing frame counter
g_Key_Has_Incom ing_Frame_Count er_c	• This indicates that the key has an incoming frame counter
g_Key_Has_Partn er_IEEE_Addr_c	This indicates that the key has an associated Partner IEEE address and the corresponding value within the ZigBeeKeyStructure_t has been populated with the data
g_Key_Is_Author ized_c	This indicates the key is authorized for use in APS data messages. If the key is not authorized for use in APS data messages it has not yet gone through a key agreement protocol, such as CBKE (i.e. ECC)

Structure: ZigBeeKeyStructure_t

Name	type	Range	Description
bitmask	ZigBeeKeyStructBitmask_t	---	This bitmask indicates the presence of information about that particular field present in bitmask.
type	Security_Key_Types	---	Type of key sent from host. It is one of key from the defined structure Security_Key_Types
key	uint8_t[16]	---	The actual value of the key to be used for Encryption and Decryption.
outgoingFrameCounter	uint32_t	0x0000 0000- 0xffffffff	This is the outgoing frame counter associated with the key. It will contain valid data based on the ZigBeeKeyStructBitmask_t.
incomingFrameCounter	uint32_t	0x0000 0000- 0xffffffff	This is the incoming frame counter associated with the key. It will contain valid data based on the ZigBeeKeyStructBitmask_t

sequenceNumber	uint8_t	0x00-0xff	This is the sequence number associated with the key.
apartnerIEEEAddress	uint8_t[8]	0x0000 0000-0xffffffff	This is the Partner IEEE Address associated with the key (Link Key)

Return Value:

On Success **:** ZigBee_Success.

On Failure : ZigBee_Invalid_Argument/ g_NO_KEY_c/ ZigBee_Failure.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.3.5 rsi_zigb_set_key_table_entry

Prototype:

```
int16_t rsi_zigb_set_key_table_entry( uint8_t index,
                          uint8_t * pIEEEAddress,
                          BOOL linkKey,
                          uint8_t * pKeyData );
```

Description:

The api allows the application to set an entry in the key table.

Parameters:

Parameters	Data type	Description
index	uint8_t	The index in the key table of the entry to set.
pIEEEAddress	uint8_t*	The address of the partner device associated with the key.
linkKey	BOOL	A boolean indicating whether this is a Link or Master Key.
pKeyData	uint8_t*	A pointer to the key data associated with the key entry.

Return Value:

On Success **:** ZigBee_Success.

On Failure : m_NO_ENTRY_c.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.3.6 rsi_zigb_add_or_update_key_table_entry

Prototype:

```
int16_t rsi_zigb_add_or_update_key_table_entry(
                                uint8_t *pIEEEAddress,
                                BOOL    linkKey,
                                uint8_t *pKeyData,
                                uint8_t *indx);
```

Description:

The api allows the application to add a new entry in the key table or updates an existing entry with a new key.

Parameters:

Parameters	Data type	Description
index	uint8_t	The index in the key table of the entry to set.
pIEEEAddress	uint8_t*	The IEEE Address of the partner device that shares the key.
linkKey	BOOL	A boolean indicating whether this is a Link or Master Key.
pKeyData	uint8_t*	A pointer to the actual key data.
indx	uint8_t	Index updated on getting response.

Return Value:

On Success **:** ZigBee_Success.

On Failure : ZigBee_Invalid_Argument/ ZigBee_Failure.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.3.7 rsi_zigb_find_key_table_entry

Prototype:

```
int16_t rsi_zigb_find_key_table_entry(uint8_t * pIEEEAddress,
                                      BOOL linkKey);
```

Description:

The api allows the application to search the key table and find an entry matching the specified IEEE address and key type.

Parameters:

Parameters	Data type	Description
pIEEEAddress	uint8_t*	The IEEE Address of the partner device that shares the key. To find the first empty entry pass in an address of all zeros.
linkKey	BOOL	A boolean indicating whether to search for an entry containing a Link or Master Key.

Return Value:

On Success **:** ZigBee_Success.

On Failure : m_NO_ENTRY_c.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.3.8 rsi_zigb_erase_key_table_entry

Prototype:

```
int16_t rsi_zigb_erase_key_table_entry(uint8_t index);
```

Description:

The api allows the application to clear a single entry in the key table.

Parameters:

Parameters	Data type	Description
index	uint8_t	The index of the trust center link key.

Return Value:

On Success **:** ZigBee_Success.

On Failure : m_NO_ENTRY_c.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.4 Binding Interface

6.4.1 rsi_zigb_set_binding_entry

Prototype:

```
typedef struct ZDP_Bind_Request_Tag {
 uint8_t          a_src_addr[8];
 uint8_t          src_endpoint;
 uint8_t          a_cluster_id[2];
 uint8_t          dest_addr_mode;
 uint8_t          a_dest_addr[8];
 uint8_t          dest_endpoint;
}ZDP_Bind_Request_t;


int16_t rsi_zigb_set_binding_entry(
                  ZDP_Bind_Request_t * pSetBindingEntry);
```

Description:

The api allows the application to set an entry in the binding table by copying the structure pointed to by pSetBindingEntry into the binding table.

Parameters:

Parameters	Data type	Description
pSetBindingEntry	ZDP_Bind_Request_t*	indicates the pointer to the binding entry which need to be set in the given index.

Structure: `ZDP_Bind_Request_t`

Name	type	Range	Description
a_src_addr	uint8_t	A valid 64-bit IEEE address	The IEEE address for the source.
src_endpoint	uint8_t	0x01-0xfe	The source endpoint for the binding entry.
a_cluster_id	uint8_t	0x0000-0xffff	The identifier of the cluster on the source device that is bound to the destination.
dest_addr_mode	uint8_t	0x00-0xff	The addressing mode for the destination address used in this command. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 – 0xff = reserved
a_dest_addr	uint8_t	As specified by the DstAddrMode field	The destination address for the binding entry.
dest_endpoint	uint8_t	0x01-0xfe	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

Return Value:

On Success **:** g_SUCCESS_c.

On Failure : g_FAILURE_c.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.4.2 rsi_zigb_get_binding_indices

Prototype:

```
int16_t rsi_zigb_get_binding_indices(uint8_t *noOfActiveIndices);
```

Description:

The api allows the application to read the active binding indices.

Parameters:

Parameters	Data type	Description
noOfActiveIndices	uint8_t*	Pointer to the list of binding indices of each uint8_t size.

Return Value:

On Success **:** ZigBee_Success.

On Failure : ZigBee_Invalid_Argument.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.4.3 rsi_zigb_delete_binding

Prototype:

```
int16_t rsi_zigb_delete_binding(uint8_t bindIndex);
```

Description:

The api allows the application to delete an entry in the binding table for the specified index.

Parameters:

Parameters	Data type	Description
bindIndex	uint8_t	Indicates the index which needs to be deleted.

Return Value:

On Success **:** ZigBee_Success.

On Failure : g_ZDP_Not_Permitted_c.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.4.4 rsi_zigb_is_binding_entry_active

Prototype:

```
int16_t rsi_zigb_is_binding_entry_active(uint8_t bindIndex);
```

Description:

The api allows the application to check whether the binding entry is active or not.

Parameters:

Parameters	Data type	Description
bindIndex	uint8_t	The index of a binding table entry.

Return Value:

On Success **:** g_TRUE_c.

On Failure : g_FALSE_c.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.4.5 rsi_zigb_clear_binding_table

Prototype:

```
int16_t rsi_zigb_clear_binding_table(void);
```

Description:

The api allows the application to clear all the binding table entries.

Parameters:

None.

Return Value:

On Success **:** ZigBee_Success.

On Failure : g_ZDP_Not_Permitted_c.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.4.6 rsi_zigb_bind_request

Prototype:

```
int16_t rsi_zigb_bind_request(
                  uint16_t shortAddress,
                  uint8_t *pIEEEAddrOfSource,
                  uint8_t sourceEndpoint,
                  uint16_t ClusterId,
                  uint8_t destAddrMode,
                  Address destAddress,
                  uint8_t destinationEndpoint,
                  BOOL APSAckRequired);
```

Description:

The api allows the application to set an entry in the binding table.

Parameters:

Parameters	Data type	Range	Description
shortAddress	uint16_t	0x0000 - 0xffff	The device short address .
ClusterId	uint16_t	0x0000-0xffff	The identifier of the cluster on the source device that is bound to the destination.
pIEEEAddrOfSource	uint8_t[8]	A valid 64-bit IEEE address	The IEEE address for the source.
sourceEndpoint	uint8_t	0x01-0xfe	The source endpoint for the binding entry.
destAddrMode	uint8_t	0x00-0xff	The addressing mode for the destination address used in this command. This field can take one of the

			non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 – 0xff = reserved
destAddress	Address	As specified by the DstAddrMode field	The destination address for the binding entry.
destinationEndpoint	uint8_t	0x01-0xfe	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.
APSAckRequired	BOOL	0x00 – 0x01	TRUE (0x00) indicates APS ack is required.

Return Value:

On Success **:** ZigBee_Success.

On Failure : ZigBee_Invalid_Argument/ ZigBee_Failure.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.4.7 rsi_zigb_unbind_request

Prototype:

```
int16_t rsi_zigb_unbind_request(
            uint16_t shortAddress,
            uint8_t *pIEEEAddrOfSource,
            uint8_t sourceEndpoint,
            uint16_t ClusterId,
            uint8_t destAddrMode,
            Address destAddress,
            uint8_t destinationEndpoint,
            BOOL APSAckRequired);
```

Description:

The api allows the application to remove bind entry between pair of device.

Parameters:

Parameters	Data type	Range	Description
shortAddress	uint16_t	0x0000 - 0xffff	The device short address .
ClusterId	uint16_t	0x0000-0xffff	The identifier of the cluster on the source device that is bound to the destination.
pIEEEAddrOfSource	uint8_t[8]	A valid 64-bit IEEE address	The IEEE address for the source.
sourceEndpoint	uint8_t	0x01-0xfe	The source endpoint for the binding entry.
destAddrMode	uint8_t	0x00-0xff	The addressing mode for the destination address used in this command. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 – 0xff = reserved
destAddress	Address	As specified by the DstAddrMode field	The destination address for the binding entry.
destinationEndpoint	uint8_t	0x01-0xfe	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.
APSAckRequired	BOOL	0x00 – 0x01	TRUE (0x00) indicates APS ack is required.

Return Value:

On Success **:** ZigBee_Success.

On Failure : ZigBee_Invalid_Argument/ ZigBee_Failure.

Returns a negative value if command issued in wrong state and packet allocation failure.

Returns -3, if command issued in wrong state

Returns -4, if packet allocation fails

If the return value is greater than 0, please refer ZigBee error code table for description.

6.5 Callbacks

6.5.1 rsi_zigb_register_callbacks

Prototype:

```
void                  rsi_zigb_register_callbacks                  (
rsi_zigb_app_scan_complete_handler_t
zigb_app_scan_complete_handler,
rsi_zigb_app_energy_scan_result_handler_t
zigb_app_energy_scan_result_handler,
rsi_zigb_app_network_found_handler_t
zigb_app_network_found_handler,
rsi_zigb_app_stack_status_handler_t
zigb_app_stack_status_handler,
rsi_zigb_app_incoming_many_to_one_route_req_handler_t
zigb_app_incoming_many_to_one_route_req_handler,
rsi_zigb_app_handle_data_indication_t
zigb_app_handle_data_indication,
rsi_zigb_app_handle_data_confirmation_t
zigb_app_handle_data_confirmation,
rsi_zigb_app_child_join_handler_t
zigb_app_child_join_handler
);
```

Description

This API used to register GAP callbacks.

Parameters

Parameter	Prototype name	Description
zigb_app_scan_complete_handler	rsi_zigb_app_scan_complete_handler_t zigb_app_scan_complete_handler	Scan complete callback
zigb_app_energy_scan_result_handler	rsi_zigb_app_energy_scan_result_handler_t zigb_app_energy_scan_result_handler	Energy Scan callback
zigb_app_network_found_handler	rsi_zigb_app_network_found_handler_t zigb_app_network_found_handler	Network Found Callback
zigb_app_stack_status_handler	rsi_zigb_app_stack_status_handler_t zigb_app_stack_status_handler	Stack status Callback
zigb_app_incoming_many_to_one_route_req_handler	rsi_zigb_app_incoming_many_to_one_route_req_handler_t zigb_app_incoming_many_to_one_route_req_handler	Route request callback
zigb_app_h	rsi_zigb_app_handle_data_i	Data indication

andle_data_indication	ndication_t zigb_app_handle_data_indication	Callback
zigb_app_handle_data_confirmation	rsi_zigb_app_handle_data_confirmation_t zigb_app_handle_data_confirmation	Data Conirmation Callback
zigb_app_child_join_handler	rsi_zigb_app_child_join_handler_t zigb_app_child_join_handler	Child join Callback

Return Values:

None

6.5.2 rsi_zigb_app_scan_complete_Handler

Prototype:

```
void r si_zigb_app_scan_complete_handler ( uint32_t channel,
uint8_t status );
```

Description:

This API is called from the stack to inform status about the status of the Current scan to the application.

Parameters:

Parameters	Data type	Description
channel	uint32_t	The channel on which the scan is enabled.
status	uint8_t	Mac status obtained would be one of the specified status in below table 5

MAC Scan Status	Value
g_MAC_Success_c	0x0
g_PAN_At_Capacity_c	0x1
g_PAN_Access_denied_c	0x2

g_MAC_Scan_In_Progress_c	0xAA
g_MAC_Beacon_Loss_c	0xE0
g_MAC_Channel_Access_Failure_c	0xE1
g_MAC_Denied_c	0xE2
g_MAC_Disable_TRX_Failure_c	0xE3
g_MAC_Failed_Security_Check_c	0xE4
g_MAC_Frame_Too_Long_c	0xE5
g_MAC_Invalid_GTS_c	0xE6
g_MAC_Invalid_Handle_c	0xE7
g_MAC_Invalid_Parameter_c	0xE8
g_MAC_No_ACK_c	0xE9
g_MAC_No_Beacon_c	0xEA
g_MAC_No_Data_c	0xEB
g_MAC_No_Short_Address_c	0xEC
g_MAC_Out_Of_CAP_c	0xED
g_MAC_PAN_ID_Conflict_c	0xEE
g_MAC_Realignment_c	0xEF
g_MAC_Transaction_Expired_c	0xF0
g_MAC_Transaction_Overflow_c	0xF1
g_MAC_TX_Active_c	0xF2
g_MAC_Unavailable_Key_c	0xF3

g_MAC_Unsupported_Attribute_c	0xF4
g_MAC_Missing_Address_c	0xF5
g_MAC_Past_Time_c	0xF6

Table : ZigBee MAC Status

6.5.3 rsi_zigb_app_energy_scan_result_handler

Prototype:

```
void rsi_zigb_app_energy_scan_result_handler( uint32_t
channel,uint8_t *pEnergyValue);
```

Description:

This API is called from the stack to report RSSI value measured on the required channel to the application.

Parameters:

Parameters	Data type	Description
channel	uint32_t	The channel on which the scan is enabled.
pEnergyValue	uint8_t*	*maxRSSIValue it's a pointer to an array of energy values in 16 channels.

6.5.4 rsi_zigb_app_network_found_handler

Prototype:

```
Struct {
      uint16_t    shortPanId,
      uint8_t     channel,
      uint8_t    extendedPanId[8],
      uint8_t    stackProfile,
      uint8_t    nwkUpdateId
      bool       allowingJoining,
      } ZigBeeNetworkDetails;


void rsi_zigb_app_network_found_handler(ZigBeeNetworkDetails
networkInformation);
```

Description:

This function is called from the application to get information about network found in the current channel.

Parameters:

Parameters	Data type	Description
networkInformation	ZigBeeNetworkDetails_t	The channel on which the scan is enabled.

Structure: **ZigBeeNetworkDetails**

Parameters	Data type	Range	Description
shortPanId	uint16_t	0x0000 - 0xffff	The network's PAN identifier
channel	uint8_t	0x0000-0xffff	The 802.15.4 channel associated with the network.
extendedPanId	uint8_t[8]	A valid 64-bit IEEE address	The network's extended PAN identifier.
stackProfile	uint8_t	0x01-0x2	The Stack Profile associated with the network
nwkUpdateId	uint8_t	0x01-0x3	The instance of the Network
allowingJoining	BOOL	0 -1	Whether the network is allowing MAC associations.

6.5.5 rsi_zigb_app_stack_status_handler

Prototype:

```
enum {
      ZigBeeNWKIsUp,
      ZigBeeNWKIsDown,
      ZigBeeJoinFailed,
      ZigBeeCannotJoinAsRouter,
      ZigBeeChangedNodeID,
      ZigBeeChangedNodeID,
      ZigBeeChangedChannel,
      ZigBeeNoBeacons,
      ZigBeeReceivedKeyInClear,
      ZigBeeNoNWKKeyReceived,
      ZigBeeNoLinkKeyReceived,
      ZigBeePreconfiguredKeyRequired,
      ZigBeeChangedManagerAddress
  } ZigBeeNWKStatusInfo;
```

```
void rsi_zigb_app_stack_status_handler(ZigBeeNWKStatusInfo
*statusInfo);
```

Description:

This callback is invoked by the ZigBee Stack to indicate any kind of Network status to the application. For example: upon establishing the network, this function shall be called by the stack to indicate status ZigBeeNetworkIsUp. If the device leaves the network, a status of ZigBeeNWkisDown status is indicated via this function call.

.

Parameters:

Parameters	Data type	Description
statusInfo	ZigBeeNWKStatusInfo _t	Stack status is one of the status mentioned in below table.

enum: `ZigBeeNWKStatusInfo`

Parameters	Description
ZigBeeNWKIsUp	indicates that Network is formed or joined successfully.
ZigBeeNWKIsDown	indicates that NWK formation failed or the device left the network.
ZigBeeJoinFailed	indicates that network join failed
ZigBeeCannotJoinAsRouter	indicates that network was unable to start as Router.
ZigBeeChangedNodeID	indicates that PANID is changed after resolving PAN ID conflict.
ZigBeeChangedChannel	indicates that the channel is changed due to frequency agility mechanism
ZigBeeReceivedKeyInClear	indicates the Network Key is received is inclear.

ZigBeeNoNWKKeyReceived	indicates no Network key is received.
ZigBeeNoLinkKeyReceived	indicates no Link key is received.
ZigBeePreconfiguredKeyRequired	indicates Preconfigured link key is required.
ZigBeeChangedManagerAddress	indicates network manager changed.

6.5.6 rsi_zigb_app_child_join_handler

Prototype:

```
void rsi_zigb_app_child_join_handler(uint16_t short_address,
                                     BOOL joining);
```

Description:

This callback is invoked is called from stack to intimate application about child device joining or leaving the network.

Parameters:

Parameters	Data type	Description
short_address	ZigBeeNWKStatusInfo _t	Child Device's short .
joining	BOOL	TRUE indicates child device joined FALSE indicates child device left network.

6.5.7 rsi_zigb_app_handle_data_confirmation

Prototype:

```
struct{
      Address dest_address;
      uint8_t dest_addr_mode;
```

```
    uint8_t dest_endpoint;
    uint8_t src_endpoint;
    uint8_t status;
}APSDE_Data_Confirmation_t;

void rsi_zigb_app_handle_data_confirmation
(APSDE_Data_Confirmation_t *pDataConfirmation);
```

Description:

This callback is invoked from stack to intimate application about child device joining or leaving the network.

Parameters:

Parameters	Data type	Description
dest_address	Address	This field indicates the individual device address or group address of the transmitted message
dest_addr_mode	uint8_t	This field indicates the destination address mode
dest_endpoint	uint8_t	This field indicates the destination endpoint to which the data frame was sent.
src_endpoint	uint8_t	This field indicates the source endpoint from which the data frame was originated.
status	uint8_t	This field indicates the status of data confirmation as shown in below Table 6.

Status	Description	Value
ZigBee_Success	No error occured while parsing the required API parameters	0x00
ZigBee_Failure	Error occured while parsing the required API parameters	0x01
ZigBee_Address_Table_Entry_Is_Active	Requested address table	0x02

	entry is active	
ZigBee_Table_Full	requested Stack table is full	0x03
ZigBee_No_Buffer	Out of buffers	0x04
ZigBee_Error_Fatal	Error occured in stack	0x05
ZigBee_Invalid_Argument	Argument passed for API is invalid	0x06
ZigBee_Fragment_Tx_Aborted	Transmission stopped inbetween in Fragmentation process	0x07
ZigBee_Fragment_Tx_Complete	Transmission Complete in Fragmentation process	0x08
ZigBee_Fragment_Rx_Aborted	Receiving stopped inbetween in Fragmentation process	0x09
ZigBee_Fragment_Reception_ Completed	Receiving Complete in Fragmentation process	0x0a
ZigBee_Fragment_Message_Too_ Long	Message Too Long	0x0b
ZigBee_Invalid_Call	Request might be not valid for the flashed device type or it is not in a state to receive call	0x0c
ZigBee_Device_Down	Device is not in network	0x0d
ZigBee_Unsupported	Feature not supported	0x0e
ZigBee_Unknown_Device_Type	Device type is unknown	0x0f
ZigBee_No_Key	No Requested Key	0x10
ZigBee_No_Entry	Entry in the table is empty	0x11
ZigBee_Index_Out_Of_Range	Accessing entry is out of range in the table	0x12
ZigBee_MAC_No_Data	No data pending	0x13
ZigBee_MAC_No_ACK	No ACK received	0x14

ZigBee_Channel_Access_Failure	MAC Channel Access Failure	0x15
ZigBee_MAC_Unavailable_Key	MAC key unavailable	0x06
ZigBee_Failed_Security_Check	MAC Failed Security Check	0x07
ZigBee_MAC_Invalid_Parameter	MAC Invalid Parameter	0x08

Table : ZigBee Data Confirmation Status

6.5.8 rsi_zigb_app_incoming_many_to_one_route_request_handler

Prototype:

```
void rsi_zigb_app_incoming_many_to_one_route_req_handler(
uint16_t SourceAddr, uint8_t * pSrcIEEEAddr,uint8_t PathCost );
```

Description:

This callback allows the Application to handle many to One Route Request

Parameters:

Parameters	Data type	Description
SourceAddr	uint16_t	The short address of the concentrator that initiated the many-to-one route request.
pSrcIEEEAddr	uint8_t*	The IEEE address of the concentrator.
PathCost	uint8_t	The path cost to the concentrator.

6.5.9 rsi_zigb_app_handle_data_indication

Prototype:

```
struct {
      Address     dest_address;
      uint8_t     dest_addr_mode;
      uint8_t     dest_endpoint;
      uint8_t     src_addr_mode;
      Address     src_address;
      uint8_t     src_endpoint;
      profile_id_t profile_id;
```

```
            cluster_id_t cluster_id;
            uint8_t      asdulength;
            uint8_t      was_broadcast;
            uint8_t      security_status;
            uint8_t      link_quality;
            uint8_t      a_asdu[1];
        } APSDE_Data_Indication_t;


void rsi_zigb_api_test_data_indication_handler(
                        APSDE_Data_Indication_t
                  *pDataIndication);
```

Description:

This callback allows the Application to handle data indication for the data request.

Parameters:

Parameters	Data type	Description
pDataIndication	APSDE_Data_Indication_t*	Contains the data indication results.

Structure: `APSDE_Data_Indication_t`

Parameters	Description
dest_address	This field the destination address in the received message.
dest_addr_mode	This field indicates the destination address mode in the receivedmessage. This field takes one of the following values: 0x00 - Indirect data transmission (destination address and destination endpoint are not present) • 0x01 - 16-bit group address • 0x02 - 16-bit address of destination device • 0x03 - 64-bit extended address of destination device • 0x04 - 0xff - Reserved
dest_endpoint	This field indicates the destination endpoint in the received message
src_addr_mode	This field indicates the source address mode in the received message
src_address	This field indicates the source address from which the message is originated
profile_id	This field indicates the 16-bit profile ID
cluster_id	This field indicates the cluster ID
Asdulength	This field indicates the length of the data received.
was_broadcast	This field indicates whether the data frame is received through broadcast
security_status	This field indicates whether the received message was secured or not and type of the security applied.
link_quality	This field indicates the LQI of the received message.
a_asdu	This field points to the actual message received

7 Appendix:

7.1 Commands and corresponding API names

The command frames specified above are represented using unique API names which are used in source code. One can find the APIs in the provided sample project. The following table provides us API name representing the corresponding command:

Command type	API name	Cmd Id
ZIGBEEFORMNETWORK	rsi_zigb_form_network	0x01
ZIGBEEJOINNETWORK	rsi_zigb_join_network	0x02
ZIGBEEPERMITJOIN	rsi_zigb_permit_join	0x03
ZIGBEELEAVENETWORK	rsi_zigb_leave_network	0x04
ZIGBEEFINDNETWORKANDPERFORMREJOIN	rsi_zigb_find_network_and_perform_rejoin	0x05
ZIGBEEREJOINNETWORK	rsi_zigb_rejoin_network	0x06
ZIGBEENETWORKRESTORE	rsi_zigb_network_restore	0x07
ZIGBEEINITIATESCAN	rsi_zigb_initiate_scan	0x08
ZIGBEESTOPSCAN	rsi_zigb_stop_scan	0x09
ZIGBEENETWORKSTATE	rsi_zigb_network_state	0x0A
ZIGBEESTACKISUP	rsi_zigb_stack_is_up	0x0B
ZIGBEEGETSELFIEEEADDRESS	rsi_zigb_get_self_ieee_address	0x0C
ZIGBEEISITSELFIEEEADDRESS	rsi_zigb_is_it_self_ieee_address	0x0D
ZIGBEEGETSELFSHORTADDRESS	rsi_zigb_get_self_short_address	0x0E
ZIGBEESETMANUFACTURERCODEFORNODEDESC	rsi_zigb_set_manufacturer_code_for_node_desc	0x0F
ZIGBEESETPOWERDESCRIPTOR	rsi_zigb_set_power_descriptor	0x10
ZIGBEESETMAXMINCOMINGTXFRSIZE	rsi_zigb_set_maxm_incoming_txfr_size	0x11
ZIGBEESETMAXMOUTGOINGTXFRSIZE	rsi_zigb_set_maxm_outgoing_txfr_size	0x12
ZIGBEESETOPERATINGCHANNEL	rsi_zigb_set_operating_channel	0x13
ZIGBEEGETDEVICETYPE	rsi_zigb_get_device_type	0x14
ZIGBEEGETOPERATINGCHANNEL	rsi_zigb_get_operating_channel	0x15
ZIGBEEGETSHORTPANID	rsi_zigb_get_short_panid	0x16
ZIGBEEGETEXTENDEDPANID	rsi_zigb_get_extended_panid	0x17

ZIGBEEGETENDPOINTID	rsi_zigb_get_endpoint_id	0x18
ZIGBEEGETSIMPLEDDESCRIPTOR	rsi_zigb_get_simple_descriptor	0x19
ZIGBEEGETENDPOINTCLUSTOR	rsi_zigb_get_endpoint_cluster	0x1A
ZIGBEEGETSHORTADDRFORSPECIFIEDIEEEADDR	rsi_zigb_get_short_addr_for_specified_ieee_addr	0x1B
ZIGBEESTACKPROFILE	rsi_zigb_stack_profile	0x1C
ZIGBEEGETIEEEADDRFORSPECIFIEDSHORTADDR	rsi_zigb_get_ieee_addr_for_specified_short_addr	0x1D
ZIGBEEREADNEIGHBOURTABLEENTRY	rsi_zigb_read_neighbor_table_entry	0x1E
ZIGBEEGETROUTETABLEENTRY	rsi_zigb_get_route_table_entry	0x1F
ZIGBEEGETREEDEPTH	rsi_zigb_tree_depth	0x20
ZIGBEEGETNEIGHBOURTABLEENTRYCOUNT	rsi_zigb_get_neighbor_table_entry_count	0x21
ZIGBEEGETCHILDSHORTADDRESSFORTHEINDEX	rsi_zigb_get_child_short_address_for_the_index	0x22
ZIGBEEGETCHILDINDEXFORSPECIFIEDSHORTADDR	rsi_zigb_get_child_index_for_specified_short_addr	0x23
ZIGBEEGETCHILDDETAILS	rsi_zigb_get_child_details	0x24
ZIGBEEENDDEVICEPOLLFORDATA	rsi_zigb_end_device_poll_for_data	0x25
ZIGBEEREADCOUNTOFCHILDDEVICES	rsi_zigb_read_count_of_child_devices	0x26
ZIGBEEREADCOUNTOFROUTERCHILDDEVICE	rsi_zigb_read_count_of_router_child_devices	0x27
ZIGBEEGETPARENTSHORTADDRESS	rsi_zigb_get_parent_short_address	0x29
ZIGBEEGETPAREANTIEEEADDRESS	rsi_zigb_get_parent_ieee_address	0x2A
ZIGBEEBROADCASTNWKMANAGERREQUEST	rsi_zigb_broadcast_nwk_manager_request	0x2C
ZDPSENDNWKADDRREQUEST	rsi_zigb_zdp_send_nwk_addr_request	0x2D
ZDPSENDIEEEADDRREQUEST	rsi_zigb_zdp_send_ieee_addr_request	0x2E
ZDPSENDDEVICEANNOUNCEMENT	rsi_zigb_zdp_send_device_announcement	0x2F
ZDPSENDMATCHDESCRIPTORSREQUEST	rsi_zigb_send_match_descriptors_request	0x30
ZIGBEEACTIVEENDPOINTSREQU	rsi_zigb_active_endpoints_request	0x31

EST		
ZDPSENDPOWERDESCRIPTORREQUEST	rsi_zigb_zdp_send_power_descriptor_request	0x32
ZDPSENDNODEDESCRIPTORREQUEST	rsi_zigb_zdp_send_node_descriptor_request	0x33
ZIGBEESIMPLEDESCRIPTORREQUEST	rsi_zigb_simple_descriptor_request	0x34
ZIGBEESENDUNICASTDATA	rsi_zigb_send_unicast_data	0x36
ZIGBEESENDGROUPDATA	rsi_zigb_send_group_data	0x37
ZIGBEEGETMAXAPSPAYLOADLENTH	rsi_zigb_get_max_aps_payload_length	0x39
ZIGBEESETBINDINGENTRY	rsi_zigb_set_binding_entry	0x3A
ZIGBEEDELETEBINDING	rsi_zigb_delete_binding	0x3B
ZIGBEEISBINDINGENTRYACTIVE	rsi_zigb_is_binding_entry_active	0x3C
ZIGBEECLEARBINDINGTABLE	rsi_zigb_clear_binding_table	0x3D
ZIGBEEBINDREQUEST	rsi_zigb_bind_request	0x3E
ZIGBEEENDDEVICEBINDREQUEST	rsi_zigb_enddevice_bind_request	0x3F
ZIGBEEUNBINDREQUEST	rsi_zigb_unbind_request	0x40
ZIGBEEGETKEY	rsi_zigb_get_key	0x41
ZIGBEEHAVELINKKEY	rsi_zigb_have_link_key	0x42
ZIGBEESWITCHNETWORKKEY	rsi_zigb_switch_network_key_handler	0x43
ZIGBEEREQUESTLINKKEY	rsi_zigb_request_link_key	0x44
ZIGBEEGETKEYTABLEENTRY	rsi_zigb_get_key_table_entry	0x45
ZIGBEESETKEYTABLEENTRY	rsi_zigb_set_key_table_entry	0x46
ZIGBEEADDORUPDATEKEYTABLEENTRY	rsi_zigb_add_or_update_key_table_entry	0x47
ZIGBEEFINDKEYTABLEENTRY	rsi_zigb_find_key_table_entry	0x48
ZIGBEEERASEKEYTABLEENTRY	rsi_zigb_erase_key_table_entry	0x49
ZIGBEESETSIMPLEDESCRIPTOR	rsi_zigb_set_simple_descriptor	0x4A
ZIGBEEGETBINDINGINDICES	rsi_zigb_get_binding_indices	0x60
ZIGBEEINITSTACK	rsi_zigb_init_stack	0x61
ZIGBEESTACKRESET	rsi_zigb_reset_stack	0x62

ZIGBEEUPDATESAS	rsi_zigb_set_tc_master_key	0x65
ZIGBEEUPDATEZDO	rsi_zigb_set_preconfigured_link_key	0x66
ZIGBEEDEINITSTACK	rsi_zigb_deinit_stack	0xFF
ZIGBEEINITPS	rsi_zigb_send_pwrmode	0x68

Table 58 Commands and API name

7.2 ZigBee status Codes

Below table shows status information of command request frames.

Status	Description	Value
ZigBee_Success	No error occured while parsing the required API parameters	0x00
ZigBee_Failure	Error occured while parsing the required API parameters	0x01
ZigBee_Address_Table_ **Entry_Is_Active**	Requested address table entry is active	0x02
ZigBee_Table_Full	requested Stack table is full	0x03
ZigBee_No_Buffer	Out of buffers	0x04
ZigBee_Error_Fatal	Error occured in stack	0x05
ZigBee_Invalid_Argument	Argument passed for API is invalid	0x06
ZigBee_Fragment_Tx_Aborted	Transmission stopped inbetween in Fragmentation process	0x07
ZigBee_Fragment_Tx_Complete	Transmission Complete in Fragmentation process	0x08
ZigBee_Fragment_Rx_Aborted	Receiving stopped inbetween in Fragmentation process	0x09
ZigBee_Fragment_Reception_ Completed	Receiving Complete in Fragmentation process	0x0a
ZigBee_Fragment_Message_Too_ Long	Message Too Long	0x0b

ZigBee_Invalid_Call	Request might be not valid for the flashed device type or it is not in a state to receive call	0x0c
ZigBee_Device_Down	Device is not in network	0x0d
ZigBee_Unsupported	Feature not supported	0x0e
ZigBee_Unknown_Device_Type	Device type is unknown	0x0f
ZigBee_No_Key	No Requested Key	0x10
ZigBee_No_Entry	Entry in the table is empty	0x11
ZigBee_Index_Out_Of_Range	Accessing entry is out of range in the table	0x12
ZigBee_MAC_No_Data	No data pending	0x13
ZigBee_MAC_No_ACK	No ACK received	0x14
ZigBee_Channel_Access_Failure	MAC Channel Access Failure	0x15
ZigBee_MAC_Unavailable_Key	MAC key unavailable	0x06
ZigBee_Failed_Security_Check	MAC Failed Security Check	0x07
ZigBee_MAC_Invalid_Parameter	MAC Invalid Parameter	0x08
ZigBee_Invalid_Cmd	Invalid Command	0xFE

Table 59 ZigBee Status Codes

Revision History

Revision No.	Version No.	Date	Author	Changes
1	1.0.10	Sep 2014	Swaraj	Initial Version
2	1.0.10fi	Oct 2014	Siddiq	Updated Commands
3	1.0.10gi	Oct 2014	Swaraj	1. Added information about ZigBee Architecture 2. Reorganized and Updated Command frames 3. Added Appendix for API names 4. Added stored configuration APIs 5. Updated API library paths
4	1.0.10i	Nov 2014	Siddiq	Added information about register command and card ready
5	1.1.0	Mar 2015	Anil	Update the structure members
6	1.2.0	June 2015	Anil	1.Corrections in Documentation 2. Added Ranges information for Command variables.
7	1.4.0	Feb 2016	Vinoth	Added SAPIs and profile content.

****