|  |  |
| --- | --- |
| NXP_logo_RGB | Application Note: JN-AN-1194c |
| ZigBee IoT Gateway - Control Bridge |
|  |  |

This Application Note describes the hardware and software components that are required to implement a ZigBee Home Automation (ZHA) Coordinator or ZigBee Light Link (ZLL) control bridge for connection to an NXP IoT Gateway host. Together these components implement a ZigBee IoT Gateway.

The ZigBee IoT Gateway Control Bridge implements the following functions:

* **ZigBee PRO Home Automation Coordinator**
* **ZigBee PRO Light Link Router**
* **IPv6 packet routing between WPAN and IoT Gateway host**
* **Wireless security using standard ZigBee PRO network layer security model**
* **Logging to IoT Gateway host syslog**

# Application Overview

The ZigBee IoT Gateway allows control and monitoring of ZigBee ZHA or ZLL devices in a ZigBee PRO network by client applications resident on devices in the IPv6 internet, allowing Zigbee devices to participate in the “Internet of Things”. This document describes the software on the JN5168 device, which forms the ZigBee IoT Gateway Control Bridge. It must be interfaced with a ZigBee IoT Gateway Host in order to form a complete ZigBee IoT Gateway. A companion document describing the ZigBee IoT Gateway Host and a User Guide describing how to update the firmware built for the Host and Control Bridge are also available in the Application Note package.

# Hardware

The required hardware to create the system described in this Application Note is:

* JN5168 device to function as an IEEE 802.15.4 WPAN interface
* Any hardware platform (PC, embedded device) capable of booting a Linux operating system (IoT Gateway host)

# Building the IoT Gateway Control Bridge

The IoT Gateway Control Bridge source files should be copied into the **workspace** directory of the JN516x ZigBee Light Link/Home Automation SDK (JN-SW-4168). This Application Note includes an Eclipse project that may be opened with the version of the Eclipse development environment provided in the ‘BeyondStudio for NXP’ toolchain (JN-SW-4141). This may be used to build the application for the JN5168 target.

Alternatively, there is a build script supplied with the Application Note, found here:

**<SDK>/workspace/JN-AN-1194-Zigbee-Gateway/Build/Build.sh**

This script may be run using the msys command line interface, from within the **Build** directory.

The application Makefile can be found here:

**<SDK>/workspace/JN-AN-1194-Zigbee-Gateway/Build/ZigbeeNodeControlBridge/**

# IoT Gateway ZigBee Control Bridge Functions

## Serial Link

The IoT Gateway host uses a serial connection via a UART to communicate with the Control Bridge. The serial link protocol is described in Appendix A.

## ZigBee Network Configuration

The Control Bridge is not preconfigured with any network settings; these are provided at runtime by the host via the serial link. Once the configuration settings such as channel number and Extended PAN ID have been sent to the node, the host specifies a run mode, which may be one of the following:

* **ZigBee HA Coordinator:** This is the default mode of operation in which the Control Bridge acts as the Coordinator for a ZigBee PRO Home Automation network. The Coordinator will allow ZigBee devices to join an unsecured network if”permit joining” is enabled (classical join); if the network is secured, the joining devices must possess the Home Automation link key. Typically this means that both HA and ZLL devices can classically join a network run in this mode
* **ZigBee Light Link Router:** In this mode the Control Bridge will allow devices to join classically as above, and will also support Touchlink joining. In a secured network, devices must possess the ZigBee Light Link certification link key (the production link key is only available to devices whose manufacturers have signed an undertaking with the ZigBee Alliance to keep the key secret). Secured networks will therefore only accept ZLL devices since HA devices do not have access to the ZLL certification link key.
* **Combined ZHA and ZLL Coordinator / Router:** This mode is a combination of the HA Coordinator and ZLL Router and is not defined by ZigBee. In this mode it is possible for ZHA and ZLL devices to classically join an unsecured or secured network, and for ZLL devices to join using Touchlinking. Joining HA and ZLL devices to the same secured network is achieved by the Coordinator/Router allowing the use of both HA and ZLL link keys.

## IPv6 Packet Routing

See also the “ZigBee IoT Gateway – Host” document which is included in the Application Note package.

The Zigbee WPAN network attached to a ZigBee IoT Gateway is mapped within an IPv6 64-bit wide (/64) address space. Any packet on the Ethernet interface of the IoT Gateway Host destined for an address with this IPv6 prefix will be routed to a virtual ZigBee device (**zb0**) in the Host. Changes to the contents of the virtual device cause commands to be sent to the appropriate ZigBee address in the WPAN from the Control Bridge.

Packets generated by a ZigBee device in the WPAN intended for an external device are delivered to the Control Bridge where the originating ZigBee address is used to map the information to the corresponding virtual device in the Host, where it becomes visible to the outside world in the application MIBs on the virtual device’s IPv6 address.

## Security

If the Host enables a secure network, all traffic in the ZigBee network layer will be encrypted using AES-128 encryption. The network key is unique to each WPAN, and is transported to devices when they join encrypted with either the ZigBee Home Automation link key or the ZigBee Light Link certification link key, using the standard procedures found in the ZigBee PRO specification.

## Logging to IoT Gateway Host Syslog

The ZigBee Control Bridge can be configured via the makefile to log either to a UART (by setting UART\_DEBUG=1) or to the Host’s syslog (by setting TRACE=1) via the serial-link. The default is to log to the Host. Messages from the Gateway Control Bridge node will appear in the syslog from ZigBeeControlBridge, prepended with the string “module: “.

Appendix A: Serial Protocol

A. Physical Characteristics

The serial link between the host processor and wireless microcontroller runs at 1Mbaud when the JN5168 is contained in a USB dongle and 921600 baud when using the serial link in the RD6040 IoT Gateway. The link settings are 8 data bits with no parity. No flow control (hardware or software) is used.

A. Message Characteristics

The protocol reserves byte values less than 0x10 for use as special characters (Start and End characters, for example). So to allow data which contains these reserved values to be sent, a procedure known as “byte stuffing” is used. This consists of identifying a byte to be sent that falls into the reserved character range, sending an Escape character (0x02) first, followed by the data byte XOR’d with 0x10.

For example, if a non-special character with the value of 0x05 is to be sent:

* Send the Escape byte (0x02)
* XOR the byte to be sent with 0x10 (0x05 xor 0x10 = 0x15)
* Send the modified byte

The messages consist of the following:

* Start character (special character)
* Message type (byte stuffed)
* Message length (byte stuffed)
* Checksum (byte stuffed)
* Message data (byte stuffed)
* End character (special character)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |  | n+6 | n+7 | n+8 |
| 0x01 |  | | n | |  |  | | | | | | 0x03 |
| Start | Msg Type | | Length | | Chksum | Data | | | | | | Stop |

Figure 1: Layout of message before byte stuffing

A. Start Character

The Start character is a single-byte special character with the value 0x01 and is sent as the first byte of any message to allow the receiving end to synchronise. Since this is considered a special character, it will be sent without modification.

A. Message Type

The message type is a 16-bit value identifying the nature of the data contained in the message payload. Values implemented are defined in the message table.

A. Message Length

The message length is a 16-bit value equal to the number of bytes in the payload section of the message, sent most significant byte first.

A. Checksum

The checksum is an 8 bit value calculated by XORing the following (starting with a checksum of 0x00):

* Message type most-significant-byte
* Message type least-significant-byte
* Message length most-significant-byte
* Message length least-significant-byte
* Data bytes

The checksum is calculated before byte stuffing the message.

A. Message Data

The message data is a number of bytes equal to the value sent as the message length field. The number of bytes transmitted via the UART may be higher due to presence of escape bytes sent to identify values that fall in the reserved range. All multi-byte binary data is sent in network byte order (big-endian).

A. End Character

The end character is a single byte special character with the value 0x03 and is sent as the last byte of any message to allow the receiving end to synchronise. Since this is considered a special character, it will be sent without modification.

A. Sequence

All commands generate a synchronous response code followed by any asynchronous responses as they become available. There is no sequence number associated with each command/response – the user must ensure that commands are issued sequentially.

Expected command response sequence:

|  |  |
| --- | --- |
| Direction | Message |
| Host -> Node | Command e.g. Get Version |
| Node -> Host | Status e.g. OK or Error, Not implemented |
| Node -> Host | Optional data messages as requested by command, e.g. Version List |

A. Data Types

The following data types are used in messages between the host and slave devices. All message definitions use 32-bit integer types, unless otherwise specified.

|  |  |
| --- | --- |
| Name | Type |
| uint8\_t | Unsigned 8 bit integer (one byte) |
| uint16\_t | Unsigned 16 bit integer (two bytes) |
| uint32\_t | Unsigned 32 bit integer (four bytes) |
| uint64\_t | Unsigned 64 bit integer (eight bytes) |
| uint128\_t | Unsigned 128 bit integer (sixteen bytes) |
| string | Buffer of characters (Variable Length, NULL Terminated) |
| data | Buffer of bytes (Variable length, calculated using message length) |

A. Response Codes

The node acknowledges each command with an “ACK” message. The message is defined in the message table.

A. ZigBee Specification

Extensive use is made of messages as defined by the ZigBee Cluster Library specification (ZigBee document 075123r04) and the ZigBee Light Link v1.0 specification (ZigBee document 11-0037-10), both of which should be used in conjunction with this document.

The ZigBee specification defines the following addressing modes which are used in the serial protocol:

|  |  |
| --- | --- |
| Address Mode | Address Mode Description |
| 0 | Bound Address |
| 1 | Group Address |
| 2 | Short Address |
| 3 | IEEE Address |

Appendix B: Serial Command Set

B. Common Commands

In the following tables, the term Node refers to the Control Bridge

B. ZigBee Stack and Node Management Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Node->Host | Status  Msg Type = 0x8000 | <status:uint8\_t>  <sequence number: uint8\_t>  <Packet Type: uint16\_t>  <Optional additional error information: string>  Status:  0 = Success  1 = Incorrect parameters  2 = Unhandled command  3 = Command failed  4 = Busy (Node is carrying out a lengthy operation and is currently unable to handle the incoming command)  5 = Stack already started (no new configuration accepted)  128 – 244 = Failed (ZigBee event codes)  Packet Type: The value of the initiating command request. | All status messages will have a sequence number sent back. Default of 0 for messages which are not transmitted over the air. |
| Node->Host | Log message  Msg Type = 0x8001 | <log level: uint8\_t>  <log message : string>  Log Level :  Use the Linux / Unix log levels  0 = Emergency  1 = Alert  2 = Critical  3 = Error  4 = Warning  5 = Notice  6 = Information  7 = Debug |  |
| Node->Host | Data Indication  Msg Type = 0x8002 | <status: uint8\_t>  <Profile ID: uint16\_t>  <cluster ID: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <source address mode: uint8\_t>  <source address: uint16\_t or uint64\_t>  <destination address mode: uint8\_t>  <destination address: uint16\_t or uint64\_t>  <payload size : uint8\_t>  <payload : data each element is uint8\_t> |  |
| Node->Host | Node Cluster List – Sent by gateway node after reset  Msg Type = 0x8003 | <source endpoint: uint8\_t t>  <profile ID: uint16\_t>  <cluster list: data each entry is uint16\_t> |  |
| Node->Host | Node Cluster Attribute List – Sent by Gateway node after reset  Msg Type = 0x8004 | <source endpoint: uint8\_t>  <profile ID: uint16\_t>  <cluster ID: uint16\_t>  <attribute list: data each entry is uint16\_t> |  |
| Node->Host | Node Command ID List – sent by Gateway node after reset  Msg Type = 0x8005 | <source endpoint: uint8\_t>  <profile ID: uint16\_t>  <cluster ID: uint16\_t>  <command ID list:data each entry is uint8\_t> |  |
| Host->Node | Get Version  Msg Type = 0x0010 | No payload | Status  Version List |
| Node->Host | Version List  Msg Type = 0x8010 | <Major version number: uint16\_t>  <Installer version number: uint16\_t> |  |
| Host->Node | Set Extended PANID  Msg Type = 0x0020 | <64-bit Extended PAN ID:uint64\_t> | Status |
| Host->Node | Set Channel Mask  Msg Type = 0x0021 | <channel mask:uint32\_t> | Status |
| Host->Node | Set Security State & Key  Msg Type = 0x0022 | <key type: uint8\_t>  <key: data> | Status |
| Host->Node | Set Device Type  Msg Type = 0x0023 | <device type: uint8\_t>  Device Types:  0 = Coordinator HA mode  1 = Router ZLL mode (pure Control Bridge)  2= Router ZLL with HA compatibility (Control Bridge with HA and ZLL security) | Status |
| Host->Node | Start Network scan  Msg Type =  0x0025 | No payload | Status  Network Joined / Formed |
| Host->Node | Start Network  Message  Type = 0x0024 | No payload | Status  Network Joined / Formed |
| Node->Host | Network Joined / Formed  Msg Type = 0x8024 | <status: uint8\_t>  <short address: uint16\_t>  <extended address:uint64\_t>  <channel: uint8\_t>  Status:  0 = Joined existing network  1 = Formed new network  128 – 244 = Failed (ZigBee event codes) |  |
| Host->Node | ZLL “Factory New” Reset  Msg Type=0x0013 | No payload  Resets (“Factory New”) the Control Bridge but persists the frame counters. | Status, followed by chip reset |
| Host->Node | “Permit join” status on the target  Msg Type = 0x0014 | No payload | Status, followed by “Permit join” status response |
| Node->Host | “Permit join” status response  Msg Type=0x8014 | <Status: bool\_t>  0 – Off  1 - On |  |
| Host->Node | Reset  Msg Type = 0x0011 | No payload | Status, followed by chip reset |
| Node->Host | Non “Factory new” Restart  Msg Type=0x8006 | Status –  0 - STARTUP  1 - WAIT\_START,  2 - NFN\_START,  3 - DISCOVERY,  4 - NETWORK\_INIT,  5 - RESCAN,  6 - RUNNING  The node is provisioned from previous restart. |  |
| Node->Host | “Factory New” Restart  Msg Type=0x8007 | Status –  0 - STARTUP  1 - WAIT\_START,  2 - NFN\_START,  3 - DISCOVERY,  4 - NETWORK\_INIT,  5 - RESCAN,  6 - RUNNING  The node is not yet provisioned. |  |
| Host->Node | Erase Persistent Data  Msg Type = 0x0012 | No payload | Status |
| Host->Node | Bind  Msg Type = 0x0030 | <target extended address: uint64\_t>  <target endpoint: uint8\_t>  <cluster ID: uint16\_t>  <destination address mode: uint8\_t>  <destination address:uint16\_t or uint64\_t>  <destination endpoint (value ignored for group address): uint8\_t> | Status  Bind response |
| Node->Host | Bind response  Msg Type = 0x8030 | <Sequence number: uint8\_t>  <status: uint8\_t> |  |
| Host->Node | Unbind  Msg Type = 0x0031 | <target extended address: uint64\_t>  <target endpoint: uint8\_t>  <cluster ID: uint16\_t>  <destination address mode: uint8\_t>  <destination address: uint16\_t or uint64\_t>  <destination endpoint(value ignored for group address): uint8\_t> | Status  Unbind response |
| Node->Host | Unbind response  Msg Type = 0x8031 | <Sequence number: uint8\_t>  <status: uint8\_t> |  |
| Node->Host | Device Announce  Msg Type = 0x004D | < short address: uint16\_t>  < IEEE address: uint64\_t>  < MAC capability: uint8\_t>  MAC capability  Bit 0 – Alternate PAN Coordinator  Bit 1 - Device Type  Bit 2 - Power source  Bit 3 - Receiver On when Idle  Bit 4,5 - Reserved  Bit 6 -Security capability  Bit 7 - Allocate Address |  |
| Host->Node | Network Address request  Msg Type = 0x0040 | <target short address: uint16\_t>  <extended address:uint64\_t>  <request type: uint8\_t>  <start index: uint8\_t>  Request Type:  0 = Single Request  1 = Extended Request | Status  Network Address response |
| Node->Host | Network Address response  Msg Type = 0x8040 | <Sequence number: uin8\_t>  <status: uint8\_t>  <IEEE address: uint64\_t>  <short address: uint16\_t>  <number of associated devices: uint8\_t>  <start index: uint8\_t>  <device list – data each entry is uint16\_t> |  |
| Host->Node | IEEE Address request  Msg Type = 0x0041 | <target short address: uint16\_t>  <short address: uint16\_t>  <request type: uint8\_t>  <start index: uint8\_t>  Request Type:  0 = Single  1 = Extended | Status  IEEE Address response |
| Node->Host | IEEE Address response  Msg Type = 0x8041 | <Sequence number: uin8\_t>  <status: uint8\_t>  <IEEE address: uint64\_t>  <short address: uint16\_t>  <number of associated devices: uint8\_t>  <start index: uint8\_t>  <device list – data each entry is uint16\_t> |  |
| Host->Node | Node Descriptor request  Msg Type = 0x0042 | <target short address: uint16\_t> | Status  Node Descriptor response |
| Node->Host | Node Descriptor response  Msg Type = 0x8042 | <Sequence number: uint8\_t>  <Status uint8\_t>  <network address: uint16\_t>  <manufacturer code: uint16\_t>  <max Rx Size: uint16\_t>  <max Tx Size: uint16\_t>  <server mask: uint16\_t>  <descriptor capability: uint8\_t>  <mac flags: uint8\_t>  <max buffer size: uint8\_t >  <bit fields: uint16\_t>  Bitfields:  Logical type (bits 0-2  0 -coordinator  1 -router  2 - ED)  Complex descriptor available (bit 3)  User descriptor available (bit 4)  Reserved (bit 5-7)  APS flags (bit 8-10 – currently 0)  Frequency band(11-15 set to 3 (2.4Ghz))  Server mask bits:  0 - Primary trust center  1 - Back up trust center  2 - Primary binding cache  3 - Backup binding cache  4 - Primary discovery cache  5 - Backup discovery cache  6 - Network manager  7 to15 - Reserved  MAC capability  Bit 0 – Alternate PAN Coordinator  Bit 1 - Device Type  Bit 2 - Power source  Bit 3 - Receiver On when Idle  Bit 4-5 - Reserved  Bit 6 - Security capability  Bit 7- Allocate Address  Descriptor capability:  0 - extended Active endpoint list available  1 - Extended simple descriptor list available  2 to 7: Reserved |  |
| Host->Node | Simple Descriptor request  Msg Type = 0x0043 | <target short address: uint16\_t>  <endpoint: uint8\_t> | Status  Simple Descriptor response |
| Node->Host | Simple Descriptor response  Msg Type= 0x8043 | <Sequence number: uint8\_t>  <status: uint8\_t>  <nwkAddress: uint16\_t>  <length: uint8\_t>  <endpoint: uint8\_t>  <profile: uint16\_t>  <device id: uint16\_t>  <bit fields: uint8\_t >  <InClusterCount: uint8\_t >  <In cluster list: data each entry is uint16\_t>  <OutClusterCount: uint8\_t>  <Out cluster list: data each entry is uint16\_t>  Bit fields:  Device version: 4 bits (bits 0-4)  Reserved: 4 bits (bits4-7) |  |
| Host->Node | Power Descriptor request  Msg Type = 0x0044 | <target short address: uint16\_t> | Status  Power Descriptor response |
| Node->Host | Power Descriptor response  Msg Type= 0x8044 | <Sequence number: uin8\_t>  <status : uint8\_t>  <bit field : uint16\_t>  Bit fields  0 to 3: current power mode  4 to 7: available power source  8 to 11: current power source  12 to15: current power source level |  |
| Host->Node | Active Endpoint request  Msg Type = 0x0045 | <target short address: uint16\_t> | Status  Active Endpoint response |
| Node->Host | Active Endpoint response  Msg Type = 0x8045 | <Sequence number: uint8\_t>  <status: uint8\_t>  <Address: uint16\_t>  <endpoint count: uint8\_t>  <active endpoint list: each data element of the type uint8\_t > |  |
| Host->Node | Match Descriptor request  Msg Type = 0x0046 | <target short address: uint16\_t>  <profile id: uint16\_t>  <number of input clusters: uint8\_t>  <input cluster list:data: each entry is uint16\_t >  <number of output clusters: uint8\_t>  <output cluster list:data: each entry is uint16\_t> | Status  Match Descriptor response |
| Node->Host | Match Descriptor response  Msg Type = 0x8046 | <Sequence number: uint8\_t>  <status: uint8\_t>  <network address: uint16\_t>  <length of list: uint8\_t>  <match list: data each entry is uint8\_t> |  |
| Host->Node | Remove Device  Msg Type = 0x0026  Management Leave request  Msg Type = 0x0047 | <target short address: uint16\_t>  <extended address: uint64\_t>  <Rejoin: uint8\_t>  <Remove Children: uint8\_t>  Rejoin,  0 = Do not rejoin  1 = Rejoin  Remove Children  0 = Leave, removing children  1 = Leave, do not remove children | Status  Management Leave response  Leave indication |
| Node->Host | Management Leave response  Msg Type = 0x8047 | <Sequence number: uin8\_t>  <status: uint8\_t> |  |
| Node->Host | Leave indication  Msg Type = 0x8048 | <extended address: uint64\_t>  <rejoin status: uint8\_t> |  |
| Host->Node | Permit Joining request  Msg Type = 0x0049 | <target short address: uint16\_t>  <interval: uint8\_t>  <TCsignificance: uint8\_t>  Target address: May be address of gateway node or broadcast (0xfffc)  Interval:  0 = Disable Joining  1 – 254 = Time in seconds to allow joins  255 = Allow all joins  TCsignificance:  0 = No change in authentication  1 = Authentication policy as spec | Status |
| Host->Node | Management Network Update request  Msg Type = 0x004A | <target short address: uint16\_t>  <channel mask: uint32\_t>  <scan duration: uint8\_t>  <scan count: uint8\_t>  <network update ID: uint8\_t>  <network manager short address: uint16\_t>  Channel Mask:  Mask of channels to scan  Scan Duration:  0 – 0xFF Multiple of superframe duration.  Scan count:  Scan repeats 0 – 5  Network Update ID:  0 – 0xFF Transaction ID for scan | Status  Management Network Update response |
| Node->Host | Management Network Update response  Msg Type = 0x804A | <Sequence number: uint8\_t>  <status: uint8\_t>  <total transmission: uint16\_t>  <transmission failures: uint16\_t>  <scanned channels: uint32\_t >  <scanned channel list count: uint8\_t>  <channel list: list each element is uint8\_t> |  |
| Host->Node | System Server Discovery request  Msg Type = 0x004B | <target short address: uint16\_t>  <Server mask: uint8\_t>  Bitmask according to spec. | Status  System Server Discovery response |
| Node->Host | System Server Discovery response  Msg Type = 0x804B | <Sequence number: uint8\_t>  <status: uint8\_t>  <Server mask: uint8\_t>  Bitmask according to spec. |  |
| Host->Node | Management LQI request  Msg Type = 0x004E | <Target Address : uint64\_t>  <Start Index : uint8\_t> | Status  Management LQI response |

B. Entire Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Node->Host | Management LQI response  Msg Type=0x804E | <Sequence number: uint8\_t>  <status: uint8\_t>  <Neighbour Table Entries : uint8\_t>  <Neighbour Table List Count : uint8\_t>  <Start Index : uint8\_t>  <List of Entries elements described below :>  NWK Address : uint16\_t  Extended PAN ID : uint64\_t  IEEE Address : uint64\_t  Depth : uint\_t  Link Quality : uint8\_t  Bit map of attributes Described below: uint8\_t  bit 0-1 Device Type  (0-Coordinator 1-Router 2-End device)  bit 2-3 Permit Join status  (1- On 0-Off)  bit 4-5 Relationship  (0-Parent 1-Child 2-Sibling)  bit 6-7 Rx On When Idle status  (1-On 0-Off) |  |
| Host->Node | Read Attribute request  Msg Type = 0x0100 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <Cluster id: uint16\_t>  <direction: uint8\_t>  <manufacturer specific: uint8\_t>  <manufacturer id: uint16\_t>  <number of attributes: uint8\_t>  <attributes list: data list of uint16\_t each>  Direction:  0 - from server to client  1 - from client to server  Manufacturer specific :  0 – No 1 – Yes | Status  Read Attribute response |
| Node->Host | Read Attribute response  Msg Type = 0x8100 | <Sequence number: uint8\_t>  <endpoint: uint8\_t>  <Cluster id: uint16\_t>  <Attribute id: uint16\_t>  <Attriibute status: uint8\_t>  <Attribute type: uint8\_t>  <Attribute value: depends on type> |  |
| Host->Node | Write Attribute request  Msg Type = 0x0110 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <Cluster id: uint16\_t>  <direction: uint8\_t>  <manufacturer specific: uint8\_t>  <manufacturer id: uint16\_t>  <number of attributes: uint8\_t>  <attributes list: data list of uint16\_t each>  Direction:  0 - from server to client  1 - from client to server  Manufacturer specific :  1 – Yes  0 – No | Data Indication  Msg Type = 0x8002 |
| Host->Node | Attribute Discovery request  Msg Type = 0x0140 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <Cluster id: uint16\_t>  <Attribute id : uint16\_t>  <direction: uint8\_t>  <manufacturer specific: uint8\_t>  <manufacturer id: uint16\_t>  <Max number of identifiers: uint8\_t>  Direction:  0 - from server to client  1 - from client to server  Manufacturer specific :  1 – Yes  0 – No | Status |
| Host->Node | Enable Permissions Controlled Joins  Msg Type = 0x0027 | <Enable/Disable : uint8\_t>  1 – Enable  2 – Disable | Status |
| Host->Node | Authenticate Device Msg Type = 0x0028 | <IEEE address ; uint64\_t>  <Key : 16 elements byte each> | Status  Authenticate response |
| Host->Node | Configure Reporting Msg Type = 0x0120 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <Cluster id: uint16\_t>  <direction: uint8\_t>  <manufacturer specific: uint8\_t>  <manufacturer id: uint16\_t>  <number of attributes: uint8\_t>  <attributes list: data list of uint16\_t each>  Direction:  0 - from server to client  1 - from client to server  Manufacturer specific :  0 – No 1 - Yes | Status |
| Node->Host | Authenticate response Msg Type = 0x8028 | <IEEE address of the Gateway: uint64\_t>  <Encrypted Key : uint8\_t 16 elements>  <MIC : uint8 4 elements>  <IEEE address of the initiating node : uint64\_t>  <Active Key Sequence number : uint8\_t>  <Channel : uint8\_t>  <Short PAN Id : uint16\_t>  <Extended PAN ID : uint64\_t> |  |
| Node->Host | Default response  Msg Type = 0x8101 | <Sequence number: uint8\_t>  <endpoint: uint8\_t>  <Cluster id: uint16\_t>  <Command Id: uint8\_t>  <Status code: uint8\_t> |  |

B. Group Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | Add Group  Msg Type = 0x0060  Command ID = 0x00 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group address: uint16\_t> | Status  Add Group response |
| Node->Host | Add Group response  Msg Type = 0x8060  Command ID = 0x00 | <Sequence number: uint8\_t>  <endpoint: uint8\_t>  <Cluster id: uint16\_t>  <status: uint8\_t>  <Group id: uint16\_t> | Status |
| Host->Node | View Group  Msg Type = 0x0061  Command ID = 0x01 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group address: uint16\_t > | Status  View Group response |
| Node->Host | View Group response  Message Type = 0x8061  Command ID = 0x01 | <Sequence number: uint8\_t>  <endpoint: uint8\_t>  <Cluster id: uint16\_t>  <status: uint8\_t>  <Group id :uint16\_t> |  |
| Host->Node | Get Group Membership  Msg Type = 0x0062  Command ID = 0x02 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group count: uint8\_t>  <group list:data> | Status  Get Group Membership response |
| Node->Host | Get Group Membership response  Msg Type = 0x8062  Command ID = 0x02 | <Sequence number: uint8\_t>  <endpoint: uint8\_t>  <Cluster id: uint16\_t>  <capacity: uint8\_t>  <Group count: uint8\_t>  <List of Group id: list each data item uint16\_t> |  |
| Host->Node | Remove Group  Msg Type = 0x0063  Command ID = 0x03 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group address: uint16\_t > | Status  Remove Group response |
| Node->Host | Remove Group response  Msg Type = 0x8063  Command ID = 0x03 | <Sequence number: uin8\_t>  <endpoint: uint8\_t>  <Cluster id: uint16\_t>  <status: uint8\_t>  <Group id: uint16\_t> | Status |
| Host->Node | Remove All Groups  Msg Type = 0x0064  Command ID = 0x04 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t> | Status |
| Host->Node | Add Group if identify  Msg Type = 0x0065  Command ID = 0x05 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group address: uint16\_t > | Status |

B. Identify Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | Identify Send  Msg Type = 0x0070 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <time: uint8\_t>  Time: Seconds | Status  Data indication |
| Host->Node | Identify Query  Msg Type = 0x0071 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t> | Status  Data indication |

B. Level Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | Move to Level  Msg Type = 0x0080 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <onoff: uint8\_t>  <mode: uint8\_t>  <rate: uint16\_t> | Status |
| Host->Node | Move to level with/without on/off  Msg Type = 0x0081 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <Level: uint8\_t >  <Transition Time: uint16\_t> | Status |
| Host->Node | Move Step  Msg Type = 0x0082 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <onoff: uint8\_t>  <step mode: uint8\_t >  <step size: uint8\_t>  <Transition Time: uint16\_t> | Status |
| Host->Node | Move Stop Move  Msg Type = 0x0083 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t> | Status |
| Host->Node | Move Stop with On Off  Msg Type = 0x0084 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t> | Status |

B. On/Off Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | On / Off with effects Send  Msg Type = 0x0094 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <effect ID: uint8\_t>  <effect gradient: uint8\_t> | Status |
| Host->Node | On/Off with no effects Msg Type = 0x0092 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <command ID: uint8\_t>  Command Id  0 – Off  1 - On  2 - Toggle | Status |
| Host->Node | On / Off Timed Send  Msg Type = 0x0093 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <onoff: uint8\_t>  <on time: uint8\_t>  <off time: uint8\_t>  On / Off:  0 = Off  1 = On  Time: Seconds | Status |

B. Scenes Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | View Scene  Msg Type = 0x00A0 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group ID: uint16\_t>  <scene ID: uint16\_t> | Status  View Scene response |
| Node->Host | View Scene response  Msg Type = 0x80A0 | <sequence number: uint8\_t>  <endpoint : uint8\_t>  <cluster id: uint16\_t>  <status: uint8\_t>  <group ID: uint16\_t>  <scene ID: uint16\_t>  <scene name length: uint8\_t>  < scene name max length: uint8\_t>  < scene name data: data each element is uint8\_t>  <extensions length: uint8\_t>  < extensions max length: uint8\_t>  < extensions data: data each element is uint8\_t> |  |
| Host->Node | Add Scene  Msg Type = 0x00A1 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group ID: uint16\_t>  <scene ID: uint16\_t>  <transition time: uint8\_t>  <scene name:string in format below>  <length: uint8\_t>  <max length: uint8\_t>  <data: data> | Status  Add Scene response |
| Node->Host | Add Scene response  Msg Type = 0x80A1 | <sequence number: uint8\_t>  <endpoint : uint8\_t>  <cluster id: uint16\_t>  <status: uint8\_t>  <group ID: uint16\_t>  <scene ID: uint16\_t> |  |
| Host->Node | Remove Scene  Msg Type = 0x00A2 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group ID: uint16\_t>  <scene ID: uint16\_t> | Status  Remove Scene response |
| Node->Host | Remove Scene response  Msg Type = 0x80A2 | <sequence number: uint8\_t>  <endpoint : uint8\_t>  <cluster id: uint16\_t>  <status: uint8\_t>  <group ID: uint16\_t>  <scene ID: uint16\_t> |  |
| Host->Node | Remove all scenes  Msg Type = 0x00A3 | <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group ID: uint16\_t> | Status  Data indication |
| Node->Host | Remove All Scene response  Msg Type = 0x80A3 | <sequence number: uint8\_t>  <endpoint : uint8\_t>  <cluster id: uint16\_t>  <status: uint8\_t>  <group ID: uint16\_t> |  |
| Host->Node | Store Scene  Msg Type = 0x00A4 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destinationendpoint: uint8\_t>  <group ID: uint16\_t>  <scene ID: uint16\_t> | Status  Data indication |
| Host->Node | Recall Scene  Msg Type = 0x00A5 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destinationendpoint: uint8\_t>  <group ID: uint16\_t>  <scene ID: uint16\_t> | Status  Data indication |
| Host->Node | Scene Membership request  Msg Type = 0x00A6 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group ID: uint16\_t> | Status  Data indication |
| Node->Host | Scene Membership response  Msg Type = 0x80A6 | <sequence number: uint8\_t>  <endpoint : uint8\_t>  <cluster id: uint16\_t>  <status: uint8\_t>  <capacity: uint8\_t>  <group ID: uint16\_t>  <scene count: uint8\_t>  <scene list: data each element uint8\_t> | Status  Data indication |

B. Colour Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | Move to Hue  Msg Type = 0x00B0 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <hue: uint32\_t>  <direction: uint8\_t>  <transition time: uint16\_t> | Status  Data indication |
| Host->Node | Move Hue  Msg Type = 0x00B1 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <mode: uint8\_t>  <rate: uint8\_t> | Status  Data indication |
| Host->Node | Step Hue  Msg Type = 0x00B2 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <mode: uint8\_t>  <step size: uint8\_t>  <transition time: uint8\_t> | Status  Data indication |
| Host->Node | Move to saturation  Msg Type = 0x00B3 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <saturation: uint8\_t>  <transition time: uint16\_t> | Status  Data indication |
| Host->Node | Move saturation  Msg Type = 0x00B4 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <mode: uint8\_t>  <rate: uint8\_t> | Status  Data indication |
| Host->Node | Step saturation  Msg Type = 0x00B5 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <mode: uint8\_t>  <step size: uint8\_t>  <transition time: uint8\_t t> | Status  Data indication |
| Host->Node | Move to hue and saturation  Msg Type = 0x00B6 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <hue: uint32\_t>  <saturation: uint32\_t>  <transition time: uint16\_t > | Status  Data indication |
| Host->Node | Move to colour  Msg Type = 0x00B7 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <colour X: uint32\_t>  <colour Y: uint32\_t>  <transition time: uint16\_t > | Status  Data indication |
| Host->Node | Move Colour  Msg Type = 0x00B8 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <colour X: uint32\_t>  <colour Y: uint32\_t>  <rate: uint8\_t> | Status  Data indication |
| Host->Node | Step Colour  Msg Type = 0x00B9 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <step X: uint8\_t>  <step Y: uint8\_t>  <transition time: uint16\_t > | Status  Data indication |

B. ZLL-specific Commands

B. Touchlink Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | Initiate Touchlink  Msg Type = 0x00D0 | No Payload | Status |
| Host->Node | Touch link factory reset target  Msg Type= 0x00D2 | No Payload | Status |
| Node->Host | Touchlink Status  Msg Type = 0x00D1 | <status: uint8\_t>  <joined node short address: uint16\_t>  Status  0 = Success  1 = Failure |  |

B. Identify Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | Identify Trigger Effect  Msg Type = 0x00E0 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <effect ID: uint8\_t>  <effect gradient: uint8\_t > | Status  Data indication |

B. On/Off Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | On / Off with Effects  Msg Type = 0x0092 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <effect ID: uint8\_t>  <effect gradient: uint8\_t> | Status  Data indication |
| Host->Node | On / Off Timed  Msg Type = 0x0093 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <onoff: uint8\_t>  <on time: uint8\_t>  <off time: uint8\_t> | Status  Data indication |

B. Scenes Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | Add Enhanced Scene  Msg Type = 0x00A7 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group ID: uint16\_t>  <scene ID: uint16\_t>  <transition time: uint8\_t>  <scene name:string>  <length: uint8\_t>  <max length: uint8\_t>  <data: data> | Status  Data indication |
| Host->Node | View Enhanced Host->Node Scene  Msg Type = 0x00A8 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <group ID: uint16\_t>  <scene ID: uint16\_t> | Status  Data indication |
| Host->Node | Copy Scene  Msg Type = 0x00A9 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <mode: uint8\_t>  <from group ID: uint16\_t>  <from scene ID: uint16\_t>  <to group ID: uint16\_t>  <to scene ID: uint16\_t> | Status  Data indication |

B. Colour Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | Enhanced Move to Hue  Msg Type = 0x00BA | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <direction: uint8\_t>  <Enhanced Hue: uint16\_t>  <transition time: uint16\_t> | Status  Data indication |
| Host->Node | Enhanced Move Hue  Msg Type = 0x00BB | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <mode: uint8\_t>  <rate: uint8\_t> | Status  Data indication |
| Host->Node | Enhanced Step Hue  Msg Type = 0x00BC | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <mode: uint8\_t>  <step size: uint8\_t>  <transition time: uint8\_t> | Status  Data indication |
| Host->Node | Enhanced Move to hue and saturation  Msg Type = 0x00BD | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <enhanced hue: uint32\_t>  <saturation: uint32\_t>  <transition time: uint8\_t> | Status  Data indication |
| Host->Node | Colour Loop Set  Msg Type = 0x00BE | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <update flags: uint8\_t>  <action: uint8\_t>  <direction: uint8\_t>  <time: uint8\_t>  <start hue: uint32\_t> | Status  Data indication |
| Host->Node | Stop Move Step  Msg Type = 0x00BF | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t> | Status  Data indication |
| Host->Node | Move to colour temperature  Msg Type = 0x00C0 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <colour temperature: uint8\_t>  <transition time: uint8\_t> | Status  Data indication |
| Host->Node | Move colour temperature  Msg Type = 0x00C1 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <mode: uint8\_t>  <rate: uint8\_t>  <minimum temperature: uint8\_t>  <maximum temperature: uint8\_t> | Status  Data indication |
| Host->Node | Step colour temperature  Msg Type = 0x00C2 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <mode: uint8\_t>  <step size: uint8\_t>  <transition time: uint8\_t>  <minimum temperature: uint8\_t>  <maximum temperature: uint8\_t> | Status  Data indication |

B. ZHA-specific Commands

B. Door Lock Cluster Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Host->Node | Lock / Unlock Door  Msg Type = 0x00F0 | <address mode: uint8\_t>  <target short address: uint16\_t>  <source endpoint: uint8\_t>  <destination endpoint: uint8\_t>  <lock/unlock: uint8\_t>  0 = Lock  1 = Unlock | Status  Data indication |

B. Exporting Persistent Data to Host

The ZigBee Control Bridge node by default uses the internal EEPROM to hold persisted data. This is about 4K on a JN5168 device and can restrict network size. To overcome this it is possible to export the data persistence to the host device. This requires a binary with this feature turned “ON”.

The host needs to provide message handshaking sequence to achieve this. How the host actually stores the persisted data is beyond the scope of the document.

|  |  |  |  |
| --- | --- | --- | --- |
| Message Direction | Message Description | Message Format | Expected Response |
| Node->Host | Host Persistent Data manager available Request  Msg Type = 0x0300 | Node enquires about the availability of the Host PDM. | Host persistent Data manager available response |
| Host->Node | Host persistent Data manager available response  Msg Type = 0x8300 | The Host must send this as the first message to allow the Node to continue operation. |  |
| Node->Host | Load Record Request Msg Type = 0x0201 | <Record Id : uint16\_t> | Load Record response |
| Host->Node | Load Record response  Msg Type = 0x8201 | <status: uint8\_t>  <Record Id: uint16\_t>  <total size: uint32\_t>  <total number of blocks: uint32\_t>  <current block: uint32\_t>  <block size: uint32\_t>  <data: variable list each item is uint8\_t>  status:   1. no record found 2. Record recovered | Status |
| Node->Host | Save Record request  Msg Type = 0x0200 | <Record Id: uint16\_t>  <total size: uint32\_t>  <total number of blocks: uint32\_t>  <current block: uint32\_t>  <block size: uint32\_t>  <data: variable list, each item is uint8\_t> | Save Record response |
| Host->Node | Save Record response  Msg Type = 0x8200 | <Record Id: uint16\_t>  <total size: uint32\_t>  <total number of blocks: uint32\_t>  <current block: uint32\_t>  <block size: uint32\_t> |  |
| Node->Host | Delete all records  Msg Type = 0x0202 |  |  |

Appendix C: Use Case Sequences

C. Gateway Start-up

The following sequence of messages is exchanged at startup. In the tables below, the Node refers to the Control Bridge

|  |  |
| --- | --- |
| Direction | Message |
| Host->Node | Erase Persistent Data (Optional) |
| Node->Host | Status (If Erase command issued) |
| Host->Node | Reset |
| Node->Host | Status |
| Node->Host | Node Cluster List (multiple) |
| Node->Host | Node Attribute List (multiple) |
| Node->Host | Node Command ID List (multiple) |
| Host->Node | Get Version |
| Node->Host | Status |
| Node->Host | Version List |
| Host->Node | Set Extended PANID |
| Node->Host | Status |
| Host->Node | Set Channel Mask |
| Node->Host | Status |
| Host->Node | Set Security State & Key |
| Node->Host | Status |
| Host->Node | Set Device Type |
| Node->Host | Status |
| Host->Node | Start Network |
| Node->Host | Status |
| Node->Host | Network Formed / Joined |

C. Touchlink Initiated by Another Control Node

|  |  |
| --- | --- |
| Direction | Message |
| Host->Node | Erase Persistent Data (Optional) |
| Node->Host | Status (If Erase command issued) |
| Host->Node | Reset |
| Node->Host | Status |
| Node->Host | Node Cluster List (multiple) |
| Node->Host | Node Attribute List (multiple) |
| Node->Host | Node Command ID List (multiple) |
| Host->Node | Get Version |
| Node->Host | Status |
| Node->Host | Version List |
| Host->Node | Set Extended PANID |
| Node->Host | Status |
| Host->Node | Set Channel Mask |
| Node->Host | Status |
| Host->Node | Set Security State & Key |
| Node->Host | Status |
| Host->Node | Set Device Type |
| Node->Host | Status |
| Host->Node | Start scan |
| Node->Host | Status |
| Node->Host | Network Joined/Failed |
| Node->Host | Touchlink status |
| Node->Host | Network formed |

C. Network Formation and Join Under Control of Host

|  |  |
| --- | --- |
| Direction | Message |
| Host->Node | Erase Persistent Data (Optional) |
| Node->Host | Status (If Erase command issued) |
| Host->Node | Reset |
| Node->Host | Status |
| Node->Host | Node Cluster List (multiple) |
| Node->Host | Node Attribute List (multiple) |
| Node->Host | Node Command ID List (multiple) |
| Host->Node | Get Version |
| Node->Host | Status |
| Node->Host | Version List |
| Host->Node | Set Extended PANID |
| Node->Host | Status |
| Host->Node | Set Channel Mask |
| Node->Host | Status |
| Host->Node | Set Security State & Key |
| Node->Host | Status |
| Host->Node | Set Device Type |
| Node->Host | Status |
| Host->Node | Start scan |
| Node->Host | Status |
| Node->Host | Network Joined/Failed |
| Host->Node | Start form |
| Node->Host | Network formed |

C. Touchlink Initiated by Host

|  |  |
| --- | --- |
| Direction | Message |
| Host->Node | Erase Persistent Data (Optional) |
| Node->Host | Status (If Erase command issued) |
| Host->Node | Reset |
| Node->Host | Status |
| Node->Host | Node Cluster List (multiple) |
| Node->Host | Node Attribute List (multiple) |
| Node->Host | Node Command ID List (multiple) |
| Host->Node | Get Version |
| Node->Host | Status |
| Node->Host | Version List |
| Host->Node | Set Extended PANID |
| Node->Host | Status |
| Host->Node | Set Channel Mask |
| Node->Host | Status |
| Host->Node | Set Security State & Key |
| Node->Host | Status |
| Host->Node | Set Device Type |
| Node->Host | Status |
| Host->Node | Start scan |
| Node->Host | Status |
| Node->Host | Network Joined/Failed |
| Host->Node | Initiate Touchlink |
| Node->Host | Touchlink status |
| Node->Host | Network formed |

C. Warm Restart

|  |  |
| --- | --- |
| Direction | Message |
| Node->Host | Warm restart status |

C. Join Notification - Device Joining Network Formed by Gateway

|  |  |
| --- | --- |
| Direction | Message |
| Node->Host | New device joined indication |
| Host->Node | Match descriptor request |
| Node->Host | Status |
| Node->Host | Match descriptor response |
| Host->Node | Add Group |
| Node->Host | Status |
| Host->Node | Identify |
| Node->Host | Status |
| Node->Host | Identify response |

C. Gateway Joins Existing Network

|  |  |
| --- | --- |
| Direction | Message |
| Host->Node | Match descriptor request (Broadcast) |
| Node->Host | Status |
| Node->Host | Match descriptor response |
| Host->Node | Add Group |
| Node->Host | Status |
| Host->Node | Identify |
| Node->Host | Status |
| Node->Host | Identify response |

C. Binding Control

No sequence required – issue Bind and Unbind commands and get status back

C. Identification

No sequence required – for HA and ZLL issue:

* Identify Send (0x0070)
* Identify Query (0x0071)

For ZLL bulbs, issue:

* Identify Trigger Effect (0x00E0)

commands and get status back

C. Scene Management

No sequence required – issue commands and get status back. For HA devices, commands are:

* View Scene (0x00A0)
* Add Scene (0x00A1)
* Remove Scene (0x00A2)
* Remove all scenes (0x00A3)
* Store Scene (0x00A4)
* Recall Scene (0x00A5)
* Scene membership request (0x00A6)

and for ZLL devices

* Add Enhanced Scene (0x00A7),
* View Enhanced Scene (0x00A8)
* Copy Scene (0x00A9)

C. Group Management

No sequence required – issue commands and get status back.

* Add Group (0x0060)
* View Group (0x0061)
* Get Group Membership (0x0062)
* Remove Group (0x0063)
* Remove All Groups (0x0064)
* Add Group if identify (0x0065)

C. On/Off Control

|  |  |
| --- | --- |
| Direction | Message |
| Host->Node | On / Off Send (0x0090) |
| Node->Host | Status |
| Node->Host | On/Off Indication |

Or

|  |  |
| --- | --- |
| Direction | Message |
| Host->Node | On / Off Timed Send (0x0091) |
| Node->Host | Status |
| Node->Host | On/Off Indication |

C. Level Control

No sequence required – issue commands and get status back

* Move to Level (0x0080)
* Move to level with/without On/Off (0x0081)
* Move Step (0x0082)
* Move Stop Move (0x0083)
* Move Stop with On/Off (0x0084)

C. Colour Control

For HA bulbs

* Move to Hue (0x00B0)
* Move Hue (0x00B1)
* Step Hue (0x00B2)
* Move to saturation (0x00B3)
* Move saturation (0x00B4)
* Step saturation (0x00B5)
* Move to hue and saturation (0x00B6)
* Move to colour(0x00B7)
* Move Colour (0x00B8)
* Step Colour (0x00B9)

For ZLL colour bulbs

* Enhanced Move to Hue (0x00BA)
* Enhanced Move Hue (0x00BB)
* Enhanced Step Hue (0x00BC)
* Enhanced Move to hue and saturation (0x00BD)
* Colour Loop Set (0x00BE)
* Stop Move Step (0x00BF)
* Move to colour temperature (0x00C0)
* Move colour temperature (0x00C1)
* Step colour temperature (0x00C2)

Revision History

|  |  |
| --- | --- |
| Version | Notes |
| 1.0 | Initial release |
| 1.1 | Package updated with extra software components |
| 2.0 | Control Bridge updated for the new ZigBee PRO stack (supplied in JN-SW-4168) and ‘BeyondStudio for NXP’ toolchain (supplied in JN-SW-4141). New commands added: Mgmt LQI, authentication, configure reporting and write attribute request, as well as support for their respective responses. Fixed bugs relating to stability and mgmt\_leave request’s rejoin and child leave features. |

Important Notice

**Limited warranty and liability —** Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors’ aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

**Right to make changes —** NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Suitability for use —** NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer’s own risk.

**Applications —** Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer’s sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer’s applications and products planned, as well as for the planned application and use of customer’s third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer’s applications or products, or the application or use by customer’s third party customer(s). Customer is responsible for doing all necessary testing for the customer’s applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer’s third party customer(s). NXP does not accept any liability in this respect.

**Export control —** This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

**Draft —** The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

All trademarks are the property of their respective owners.

|  |  |  |
| --- | --- | --- |
|  | **NXP Semiconductors**  For the contact details of your local NXP office or distributor, refer to: |  |
|  | [**www.nxp.com**](http://www.nxp.com) |  |