Z-Home – Zigbee

Technical Functional Specification

DRAFT

Author: Andrea Ravalli

Date: 12/12/2016

Revision: 0.3

Table of Contents

[1. Definitions 7](#_Toc474066488)

[1.1 Addresses 7](#_Toc474066489)

[1.2 Other definitions 7](#_Toc474066490)

[1.3 Frame Types 7](#_Toc474066491)

[1.3.1 TX\_EXPLICIT 7](#_Toc474066492)

[1.3.2 RX\_EXPLICIT 7](#_Toc474066493)

[2 Xbee module configuration 8](#_Toc474066494)

[2.1 Settings 8](#_Toc474066495)

[2.2 AT command sequence 8](#_Toc474066496)

[3 General Concepts 9](#_Toc474066497)

[3.1 Xbee explicit frames and APS frames 9](#_Toc474066498)

[3.2 Reading ZDO message 10](#_Toc474066499)

[4 Nodes and Service Discovery 10](#_Toc474066500)

[4.1 Route Record Request 10](#_Toc474066501)

[4.1.1 Description 10](#_Toc474066502)

[4.1.2 Pseudo code 10](#_Toc474066503)

[4.1.3 Frame 11](#_Toc474066504)

[4.2 Route Record Response 11](#_Toc474066505)

[4.2.1 Description 11](#_Toc474066506)

[4.2.2 Pseudo code 12](#_Toc474066507)

[4.2.3 Frame 12](#_Toc474066508)

[4.3 Active Endpoint Request 13](#_Toc474066509)

[4.3.1 Description 13](#_Toc474066510)

[4.3.2 Pseudo code 13](#_Toc474066511)

[4.3.3 Frame 13](#_Toc474066512)

[4.4 Active Endpoint Response 14](#_Toc474066513)

[4.4.1 Description 14](#_Toc474066514)

[4.4.2 Pseudo code 14](#_Toc474066515)

[4.4.3 Frame 14](#_Toc474066516)

[4.5 Simple Descriptor Request 15](#_Toc474066517)

[4.5.1 Description 15](#_Toc474066518)

[4.5.2 Pseudo code 16](#_Toc474066519)

[4.5.3 Frame 16](#_Toc474066520)

[4.6 Simple Descriptor Response 17](#_Toc474066521)

[4.6.1 Description 17](#_Toc474066522)

[4.6.2 Pseudo code 17](#_Toc474066523)

[4.6.3 Frame 17](#_Toc474066524)

[4.7 Device Announce Message 18](#_Toc474066525)

[4.7.1 Description 19](#_Toc474066526)

[4.7.2 Pseudo code 19](#_Toc474066527)

[4.7.3 Frame 19](#_Toc474066528)

[5 Nodes discovery: startup sequence and persistency 20](#_Toc474066529)

[5.1 Psedocode 20](#_Toc474066530)

[5.2 Sequence Diagram 20](#_Toc474066531)

[5.3 Data Types (proposal) 20](#_Toc474066532)

[6 Intruder Alarm System Devices 21](#_Toc474066533)

[6.1 Enrolment Sequence 21](#_Toc474066534)

[6.1.1 Service discovery 21](#_Toc474066535)

[6.1.2 Enrollment sequence 22](#_Toc474066536)

[6.2 Commands and attributes 23](#_Toc474066537)

[6.2.1 Enroll Request 24](#_Toc474066538)

[6.2.2 Enroll Response 24](#_Toc474066539)

[6.2.3 Write attribute (IAS\_CIE\_Address) 25](#_Toc474066540)

[6.2.4 Write attribute response 25](#_Toc474066541)

[6.2.5 Read attribute (Zone Status) 25](#_Toc474066542)

[6.2.6 Read attribute response 25](#_Toc474066543)

[6.2.7 Zone Status Change 25](#_Toc474066544)

[6.2.8 IAS\_CIE\_Address attribute 25](#_Toc474066545)

[6.2.9 Zone Status attribute 25](#_Toc474066546)

[6.2.10 Zone Status attribute 25](#_Toc474066547)

**Table of Figures**

[Figure 1: This figure shows which building block is responsible for the implementation of each Zigbee protocol stack element. 5](#_Toc474067345)

[Figure 2: In this figure are highlighted the offered services for each Zigbee protocol stack element. 6](#_Toc474067346)

[Figure 3: Enrollment sequence triggered the Zone device after joining the network. In this scenario the initiator is the Zone device searching (Matching\_Desc\_req) for a CIE device in the network. 22](file:///C:\Users\ravallian\git\ZigbeeTester\Docs\TFS_Z.Home_NTTDATA.docx#_Toc474067347)

[Figure 4: Enrollment sequence triggered by the CIE writing the IAS\_CIE\_Address attribute of the Zone device. In this scenario the initiator is the CIE discovering a new Zone device in the network. 23](file:///C:\Users\ravallian\git\ZigbeeTester\Docs\TFS_Z.Home_NTTDATA.docx#_Toc474067348)

# Zigbee protocol stack

Below are reportet two figures depiting the Zigbee protocol stack organization with reference to the OSI model and to the offered services.

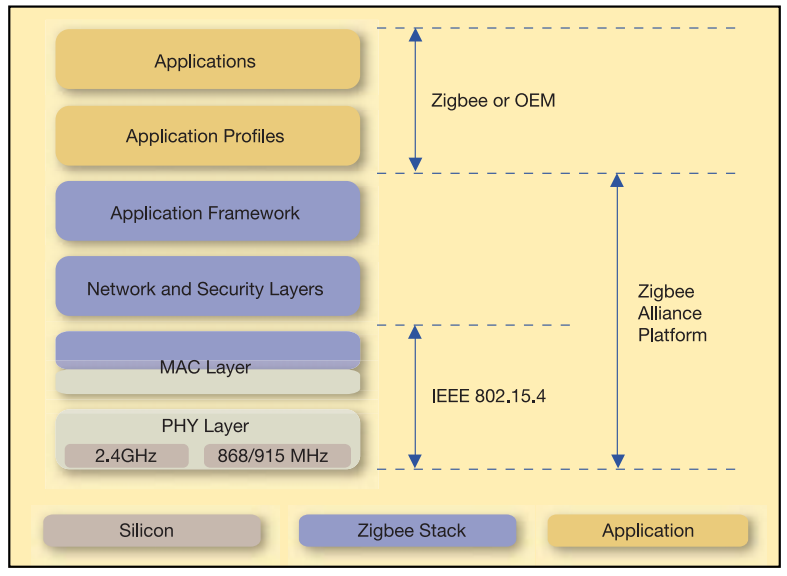


Figure 1: This figure shows which building block is responsible for the implementation of each Zigbee protocol stack element.

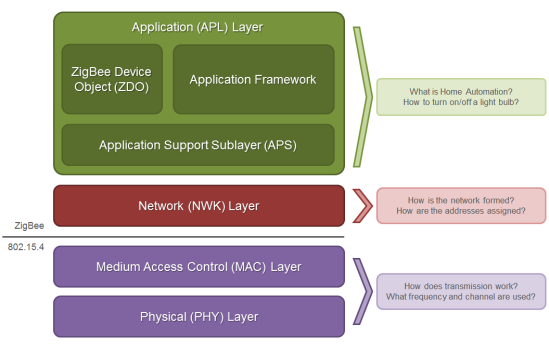


Figure 2: In this figure are highlighted the offered services for each Zigbee protocol stack element.

# Definitions

Below are reported the definitions used in this document.

## Addresses

* 64bit address BROADCAST:  
  0x000000000000FFFF
* 16bit address UNKNOWN (BROADCAST):  
  0xFFFE
* DEV\_ADDR\_64: 64 bit address for a known device (already added to the associated device map):  
  0xXXXXXXXXXXXXXXXX
* DEV\_ADDR\_16: 16 bit address for a known device (already added to the associated device map):

0xYYYY

## Other definitions

* Unspecified values or to be calculated are marked by dashes:  
  1 byte to be calculated ‘0x—‘

## Frame Types

### TX\_EXPLICIT

Il *Explicit Addressing Command frame* sarà indicato brevemente come ***tx\_explicit.***

Esso permette di inviare frame di comando esplicitantando anche gli indirizzi di destinazione. Richiede API Output (AO) mode settato ad 1 o superiore.

Per dettagli vedere pag 161 di Xbee\_S2C\_manual\_90002002.pdf.

### RX\_EXPLICIT

Il *Explicit Rx Indicator frame* sarà indicato brevemente come ***rx\_explicit.***

Rappresenta la risposta ad un frame di comando di tipo ***tx\_explicit*** con ***AO>=1***.

Per dettagli vedere pag 176 di Xbee\_S2C\_manual\_90002002.pdf

# Xbee module configuration

## Settings

Zigbee Coordinator (CE) 1

Device Join Permit (JN) FF

Zigbee Stack Profile (ZS) 2

Encryption Enable (EE) 1

Encryption Options (EO) 0

Encryption Key (KY) 5a6967426565416c6c69616e63653039

Network Encryption Key (NK) 0

API Enable (AP) 1

API Output Mode (AO) 3

Sleep Mode (SM) 1

## AT command sequence

Below is reported the list of AT commands as they were sent to the XBee device in “transparent mode”, ordered such to avoid command conflicts/device inconsistent states. The same commands can be sent in “API mode” using the “AT command frame”

The string ‘ERROR’ is reported on the serial output in case of write failures.

The string ‘OK’ is reported …

+++

ATSM 1

ATCE 1

ATJN FF

ATZS 2

ATEE 1

ATEO 0

ATKY 5a6967426565416c6c69616e63653039

ATNK 0

ATAP 1

ATAO 3

ATWR

ATFR

# General Concepts

## Xbee explicit frames and APS frames

The Xbee firmware will abstract all the Zigbee protocol levels up to the APS level.



Figure 3: Zigbee stack in detail.

APS frames are encapsulated inside the Xbee explicit frame in the “payload” field. This field is sent as is over the air and since that has to written in “little endian” byte order.



Figure 4: Zigbee stack with frames envelopments.

## Reading ZDO message

In order to read ZDO messages the Xbee AO (API Output mode) needs to be set to 3.

ZDO messages needs to be formatted as APS frames including the APS header (frame type and transaction id included). Detail about the APS frames can be found in ZigbeeSpec  cap 2.4.2.8.

# Nodes and Service Discovery

This chapter will list all ZDO frames useful to the construction of the coordinator’s map all the devices belonging to his network altogether with the services they offer.

## Route Record Request

* **Cluster ID: 0x0032**
* **Profile ID: 0x0000**

### Description

Il coordinator al suo avvio controlla quali dispositivi appartengono alla sua rete chiedendo i il contenuto della loro routing table ai dispositivi connessi.

Questa azione può (deve?) essere usata per costrure la tabella dei dispositivi connessi.

Details are available at page 12 of APP\_NOTE\_XBee\_ZigBee\_Device\_Profile.pdf

### Pseudo code

zb.send('tx\_explicit',

dest\_addr\_long = BROADCAST,

dest\_addr = UNKNOWN,

src\_endpoint = '\x00',

dest\_endpoint = '\x00',

cluster = '\x00\x32',

profile = '\x00\x00',

data = '\x12'+'\x01'

)

### Frame

Frame details:

|  |  |  |
| --- | --- | --- |
| Frame data fields | Offset | Value |
| Start delimiter | 0 | 0x7e |
| Length | MSB 1 - LSB 2 | 0x0015 |
| Frame Type | 3 | 0x11 |
| Frame Seq ID | 4 | 0xAA |
| 64-bit dest addr | MSB 5 - LSB 12 | BROADCAST |
| 16-bit dest addr | MSB 13 - LSB 14 | UNKNOWN |
| src end point | 15 | 0x00 |
| dest end point | 16 | 0x00 |
| cluster ID | 17-18 | 0x0032 |
| Profile ID | 19-20 | 0x0000 |
| broadcast radius | 21 | 0x00 |
| tx options | 22 | 0x00 |
| Data Payload | 23-24 | 0x12 0x01 |
| Checksum | 25 | -- |

Payload Data description

|  |  |  |
| --- | --- | --- |
| data fields | Offset | Example |
| Transaction ID | 0 | 0x12 |
| Start Index | 1 | 0x01 |

Hexadecimal representation:

0x76001511AA000000000000FFFFFFFE00000032000000001201--

## Route Record Response

* **Cluster ID: 0x8032**
* **Profile ID: 0x0000**

### Description

This message is the reply to a Route Record Request and his payload will report the routing table of the answering device if supported, otherwise a CLUSTER\_UNSUPP error code will returned.

Since this message can be received multiple times (if the request was sent in broadcast) even from device not supporting this cluster (typically ZED does not support it), it can be used as alternate strategy for discovering all nodes joined to a network (only the awaken devices will reply).

Details are available at page 12 of APP\_NOTE\_XBee\_ZigBee\_Device\_Profile.pdf

### Pseudo code

N/A.

### Frame

Frame details:

|  |  |  |
| --- | --- | --- |
| Frame data fields | Offset | Value |
| Start delimiter | 0 | 0x7e |
| Length | MSB 1 - LSB 2 | 0x00-- |
| Frame Type | 3 | 0x91 |
| 64-bit dest addr | MSB 4 - LSB 11 | DEV\_ADDR\_64 |
| 16-bit dest addr | MSB 12 - LSB 13 | DEV\_ADDR\_16 |
| src end point | 14 | 0x00 |
| dest end point | 15 | 0x00 |
| cluster ID | 16-17 | 0x8032 |
| Profile ID | 18-19 | 0x0000 |
| rx options | 20 | 0x-- |
| Data Payload | 21-n | -- |
| Checksum | n+1 | -- |

Data Payload description:

|  |  |  |
| --- | --- | --- |
| data fields | Offset | Example |
| Transaction ID | 0 | 0x12 |
| Routing Table | 1 | 0x01 |

Routing Table:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Size (bytes)** | **Description** |
| Routing Table Entries | 1 |  |
| Routing Table Entries | 1 | The total number of routing table entries |
| Start Index | 1 | The starting point in the routing table |
| Routing Table List Count | 1 | The number of routing table entries in this response |
| Routing Table List | Variable | A list of routing table entries. |

For the Routing Table List details see p.12 di APP\_NOTE\_XBee\_ZigBee\_Device\_Profile.pdf.

Hexadecimal representation: N/A

## Active Endpoint Request

* **Cluster ID: 0x0005**
* **Profile ID: 0x0000**

### Description

Transmission used to discover the active endpoints on a device with a matching 16-bit address.

This action is part of the service discovery process.

Details are available at page 7 of APP\_NOTE\_XBee\_ZigBee\_Device\_Profile.pdf

### Pseudo code

zb.send('tx\_explicit',

dest\_addr\_long = DEV\_ADDR\_64,

dest\_addr = DEV\_ADDR\_16,

src\_endpoint = '\x00',

dest\_endpoint = '\x00',

cluster = '\x00\x05',

profile = '\x00\x00',

data = DEV\_ADDR\_16 [1]+ DEV\_ADDR\_16 [0]

)

### Frame

Frame details:

|  |  |  |
| --- | --- | --- |
| Frame data fields | Offset | Value |
| Start delimiter | 0 | 0x7e |
| Length | MSB 1 - LSB 2 | 0x0015 |
| Frame Type | 3 | 0x11 |
| Frame Seq ID | 4 | 0x-- |
| 64-bit dest addr | MSB 5 - LSB 12 | DEV\_ADDR\_64 |
| 16-bit dest addr | MSB 13 - LSB 14 | DEV\_ADDR\_16 |
| src end point | 15 | 0x00 |
| dest end point | 16 | 0x00 |
| cluster ID | 17-18 | 0x0005 |
| Profile ID | 19-20 | 0x0000 |
| broadcast radius | 21 | 0x00 |
| tx options | 22 | 0x00 |
| Data Payload | MSB 23 - LSB 24 | DEV\_ADDR\_16 |
| Checksum | 25 | -- |

Payload Data description

|  |  |  |
| --- | --- | --- |
| data fields | Offset | Example |
| Transaction ID | 0 | 0x12 |
| Device addres | 1 | DEV\_ADDR\_16 |

Hexadecimal representation:

0x76001511-- XXXXXXXXXXXXXXXXYYYY0000000500000000YYYY--

## Active Endpoint Response

* **Cluster ID: 0x8005**
* **Profile ID: 0x0000**

### Description

Indicates the list of active endpoints supported on the device.

Gli end point riportati devono essere aggiunti alla tabella di mapping del dispositivo al fine di poter essere successivamente utlilizzati.

Details are available at APP\_NOTE\_XBee\_ZigBee\_Device\_Profile.pdf page 7.

### Pseudo code

N/A.

### Frame

Frame details:

|  |  |  |
| --- | --- | --- |
| Frame data fields | Offset | Value |
| Start delimiter | 0 | 0x7e |
| Length | MSB 1 - LSB 2 | 0x00-- |
| Frame Type | 3 | 0x91 |
| 64-bit dest addr | MSB 4 - LSB 11 | DEV\_ADDR\_64 |
| 16-bit dest addr | MSB 12 - LSB 13 | DEV\_ADDR\_16 |
| src end point | 14 | 0x00 |
| dest end point | 15 | 0x00 |
| cluster ID | 16-17 | 0x8005 |
| Profile ID | 18-19 | 0x0000 |
| rx options | 20 | 0x-- |
| Data Payload | 21-n | -- |
| Checksum | n+1 | -- |

Data Payload description

|  |  |  |
| --- | --- | --- |
| data fields | Offset | Example |
| Transaction ID | 0 | 0x12 |
| Active End points | 1 | -- |

Active End points list:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Size (bytes)** | **Description** |
| Status | 1 |  |
| Network Address | 2 | Indicates the 16-bit address of the responding device |
| Active Endpoint Count | 1 | Number of endpoints in the following endpoint list |
| Active Endpoint List | n | List of endpoints supported on the destination device. One byte per |

Data payload description extracted from APP\_NOTE\_XBee\_ZigBee\_Device\_Profile.pdf p.7.

Hexadecimal representation: N/A

## Simple Descriptor Request

* **Cluster ID: 0x0004**
* **Profile ID: 0x0000**

### Description

Transmission used to discover the active endpoints on a device with a matching 16-bit address.

Questa azione deve essere usata per costrure la mappa degli end\_point disponibili per ciascuno dei dispositivi connessi.

Per dettagli vedere pag 6 di APP\_NOTE\_XBee\_ZigBee\_Device\_Profile.pdf

### Pseudo code

zb.send('tx\_explicit',

dest\_addr\_long = data['source\_addr\_long'],

dest\_addr = data['source\_addr'],

src\_endpoint = '\x00',

dest\_endpoint = '\x00',

cluster = '\x00\x04',

profile = '\x00\x00',

data = '\x13' + data['source\_addr'][1] + data['source\_addr'][0] + '\x01'

)

### Frame

Frame details:

|  |  |  |
| --- | --- | --- |
| Frame data fields | Offset | Value |
| Start delimiter | 0 | 0x7e |
| Length | MSB 1 - LSB 2 | 0x0016 |
| Frame Type | 3 | 0x11 |
| Frame Seq ID | 4 | 0x-- |
| 64-bit dest addr | MSB 5 - LSB 12 | DEV\_ADDR\_64 |
| 16-bit dest addr | MSB 13 - LSB 14 | DEV\_ADDR\_16 |
| src end point | 15 | 0x00 |
| dest end point | 16 | 0x00 |
| cluster ID | 17-18 | 0x0004 |
| Profile ID | 19-20 | 0x0000 |
| broadcast radius | 21 | 0x00 |
| tx options | 22 | 0x00 |
| Data Payload | 23-25 | YYYY01 |
| Checksum | 26 | -- |

Payload Data description

|  |  |  |
| --- | --- | --- |
| data fields | Offset | Example |
| Transaction ID | 0 | -- |
| Device addres | 1 | DEV\_ADDR\_16 |
| Target End Point | 3 | 0x1 |

Hexadecimal representation:

0x76001511--XXXXXXXXXXXXXXXXYYYY0000000400000000YYYY01--

## Simple Descriptor Response

* **Cluster ID: 0x8004**
* **Profile ID: 0x0000**

### Description

This message will report the simple descriptor of the device. This descriptor contenis all the useful details related to the queried end point, included the device ID, defining the device class and purpose, and the available clusters on it.

Details are available at page 6 of APP\_NOTE\_XBee\_ZigBee\_Device\_Profile.pdf

### Pseudo code

N/A.

### Frame

Frame details:

|  |  |  |
| --- | --- | --- |
| Frame data fields | Offset | Value |
| Start delimiter | 0 | 0x7e |
| Length | MSB 1 - LSB 2 | 0x00-- |
| Frame Type | 3 | 0x91 |
| 64-bit dest addr | MSB 4 - LSB 11 | DEV\_ADDR\_64 |
| 16-bit dest addr | MSB 12 - LSB 13 | DEV\_ADDR\_16 |
| src end point | 14 | 0x00 |
| dest end point | 15 | 0x01 |
| cluster ID | 16-17 | 0x8005 |
| Profile ID | 18-19 | 0x0000 |
| rx options | 20 | 0x-- |
| Data Payload | 21-n | -- |
| Checksum | n+1 | -- |

Data Payload description

|  |  |  |
| --- | --- | --- |
| data fields | Offset | Example |
| Transaction ID | 0 | 0x12 |
| Descriptor | 1 | -- |

Descriptor:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Size (bytes)** | **Description** |
| Status | 1 |  |
| Network Address | 2 | Indicates the 16-bit address of the responding device |
| Length | 1 | Length of the simple descriptor |
| Simple Descriptor | Variable | See simple descriptor below. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | **Size (bits)** | **Description** | |
| Endpoint | | 8 | The endpoint on the node to which this descriptor refers. | |
| Application profile ID | | 16 | The profile ID supported on this endpoint. | |
| Application device ID | | 16 | Specifies the device description identifier supported on the device | |
| Application device version | | 4 | The version of the device description supported on this endpoint. | |
| Reserved | | 4 | |
| Input cluster count | 8 | | The number of input clusters supported on this endpoint. | |
| Input cluster list | Variable | | The list of input clusters supported on this endpoint. Each cluster is 2 bytes in size. This field is not included if the input cluster count is 0. | |
| Output cluster count | 8 | | The number of output clusters supported on this endpoint. | |
| Output cluster list | Variable | | The list of output clusters supported on this endpoint. Each cluster is 2 bytes in size. This field is not included if the output cluster count is 0. | |

Data payload description extracted from APP\_NOTE\_XBee\_ZigBee\_Device\_Profile.pdf p.6.

Hexadecimal representation: N/A

## Device Announce Message

* **Cluster ID: 0x0013**
* **Profile ID: 0x0000**

### Description

Sent by devices announcing them self in the network.

This devices need to be added to the associated device map.

### Pseudo code

N/A.

### Frame

Frame details:

|  |  |  |
| --- | --- | --- |
| Frame data fields | Offset | Value |
| Start delimiter | 0 | 0x7e |
| Length | MSB 1 - LSB 2 | 0x00-- |
| Frame Type | 3 | 0x91 |
| 64-bit dest addr | MSB 4 - LSB 11 | DEV\_ADDR\_64 |
| 16-bit dest addr | MSB 12 - LSB 13 | DEV\_ADDR\_16 |
| src end point | 14 | 0x00 |
| dest end point | 15 | 0x00 |
| cluster ID | 16-17 | 0x0013 |
| Profile ID | 18-19 | 0x0000 |
| rx options | 20 | 0x-- |
| Data Payload | 21 | -- |
| Checksum | n+1 | -- |

Data Payload description

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Size (bytes)** | **Description** |
| Transaction ID | 1 |  |
| Capabilities | 1 |  |

Hexadecimal representation: N/A

# Nodes discovery: startup sequence and persistency

At boot time the device should load from the Xbee persistency the list of associated devices. These data are stored by Xbee module in its persistency and can be retrieved by requesting the neighbor table (ZDO cluster 0x0032) or the child table (extended request for ZDO cluster 0x0000 and 0x0001).

The information retrieved should be stored set of suitable data structure or DB.

## Psedocode

if starting

send: neighbor\_table\_req(coordinator)

while: running

if new\_message

goto: parse\_msg

print: dev\_map

parse\_msg:

if neighbor\_table\_res or device\_announce:

add discovered\_device to device\_map

for each dev in device\_map do:

send: active\_endpoint\_req

else if active\_end\_point\_res:

build end\_point\_map

link end\_point\_map to device\_entry in device\_map

for each end\_point in end\_point\_map do:

send: simple\_descriptor\_req

else if simple\_descriptor\_re:

add: descriptor to end\_point in end\_point\_map

## Sequence Diagram

TBD

## Data Types (proposal)



# Intruder Alarm System Devices

On a Zigbee network a special regards is dedicated to the security devoted devices and are managed under the IAS, Intrusion and Alarm System, set.

The IAS devices are classified in: Control Indicating Equipment (CIE), Zones device, Warning Device (WD) and Ancillary Control Equipment (ACE).

CIE devices are in charge to collect all the alarms coming from Zone devices and they will implements all the related business logic. For instance, in case it’s available in the network, it will activate the WD device.

Others details about the IAS device family (and related clusters) are available in chapter 8 of ZCL.

We will take care only of the CIE and Zones device: the CIE is our network coordinator and the Zones devices are the managed sensors. They are described in chapter 8 of the ZCL and in chapters 7 and 10.7 through 10.9.

The Home Automation profile extends the ZCL definition IAS cluster’s attributes and commands.

## Enrolment Sequence

An IAS Zone device joining a new network needs to know who is the CIE for that network and in case there is more than just one, which one is controlling his zone. In other term it needs to know to whom has to send its alarms.

This task is split in two parts: the service discovery and the enrollment sequence.

### Service discovery

The service discovery is initiated by the IAS Zone devices immediately after joining a network by sending a Matching\_Desc\_req message and waiting for a response.

The request is sent in broadcast and is repeated for a few times in intervals of a few seconds (for the Heinman Water Leakage sensor it’s repeated 11 times in intervals of 9 seconds).

If a successful Matching\_Desc\_res is received the IAS Zone device will start the enrollment sequence described in the next chapter.

If after the max number of attempts the IAS Zone will not receive a Matching\_Desc\_res it will enter in an “idle” state waiting for commands.

That is the case when the CIE device is joining the network after the Zone device.

In this case the CIE will perform a service discovery by using the ZDO commands described in 4.5 and once it found an IAS Zone device (device ID 0x0402), it can trigger the enrollment sequence by explicitly writing its IEEE address in the IAS\_CIE\_Address attribute of the Zone device.

### Enrollment sequence

With the enrolment sequence the Zone device require the CIE to take in charge its zone. If the CIE addressed by the request (more than one CIE is possible in the network) has interest for the requiring Zone (there are several reason for a CIE to discard a Zone) will it a zone (will beaded to a zone in the zones table) and then send a success message with the assigned zone id.

Below are reported the sequence for the enrollment in the two scenarios where, first, the initiator is the Zone device and, second, the initiator is the CIE.

IAS CIE Device

IAS Zone Device

The device has join the network

Device Announce

Matching\_Desc\_req

Matching\_Desc\_res

Zone\_Enroll\_req

Zone\_Enroll\_res

Figure 5: Enrollment sequence triggered the Zone device after joining the network. In this scenario the initiator is the Zone device searching (Matching\_Desc\_req) for a CIE device in the network.

IAS CIE Device

IAS Zone Device

The device has just discovered an IAS Zone devive in the network

write: IAS\_CIE\_Address

write success

Zone\_Enroll\_req

Zone\_Enroll\_res

Figure 6: Enrollment sequence triggered by the CIE writing the IAS\_CIE\_Address attribute of the Zone device. In this scenario the initiator is the CIE discovering a new Zone device in the network.

## Commands and attributes

Cleared that the “server” role is kept for the device owning the cluster attributes, for the IAS Zone cluster (0x0500) the server is the Zone device (ZED) and the client is the CIE (coordinator or router).

In the next chapters will be described all the mentioned attributes and the commands.

### Enroll Request

This is a server side unicast request. The XBee module will present it as an “RX\_EXPLICIT”.

Frame details:

|  |  |  |
| --- | --- | --- |
| Frame data fields | Offset | Value |
| Start delimiter | 0 | 0x7e |
| Length | MSB 1 - LSB 2 | 0x00-- |
| Frame Type | 3 | 0x91 |
| Frame Seq ID | 4 | 0x-- |
| 64-bit dest addr | MSB 5 - LSB 12 | CIE\_ADDR\_64 |
| 16-bit dest addr | MSB 13 - LSB 14 | CIE\_ADDR\_16 |
| src end point | 15 | 0x01 |
| dest end point | 16 | 0xXX |
| cluster ID | 17-18 | 0x0500 |
| Profile ID | 19-20 | 0x0104 |
| broadcast radius | 21 | 0x00 |
| tx options | 22 | 0x00 |
| Data Payload | 23-25 | YYYY |
| Checksum | 26 | -- |

Data Payload description:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Size (bytes)** | **Description** |
| Transaction ID | 1 |  |
|  |  |  |

### Enroll Response

This is a client side unicast response. It will be sent through the XBee module as a “TX\_EXPLICIT”.

Frame details:

|  |  |  |
| --- | --- | --- |
| Frame data fields | Offset | Value |
| Start delimiter | 0 | 0x7e |
| Length | MSB 1 - LSB 2 | 0x00-- |
| Frame Type | 3 | 0x11 |
| Frame Seq ID | 4 | 0x-- |
| 64-bit dest addr | MSB 5 - LSB 12 | DEV\_ADDR\_64 |
| 16-bit dest addr | MSB 13 - LSB 14 | DEV\_ADDR\_16 |
| src end point | 15 | 0x01 |
| dest end point | 16 | 0xXX |
| cluster ID | 17-18 | 0x0500 |
| Profile ID | 19-20 | 0x0104 |
| broadcast radius | 21 | 0x00 |
| tx options | 22 | 0x00 |
| Data Payload | 23-28 | -- |
| Checksum | 29 | -- |

Data Payload description:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Size (bytes)** | **Description** |
| Frame Type | 1 |  |
| Transaction ID | 1 |  |
| Command | 1 |  |
| Enrolment Status | 1 |  |
| Zone ID | 1 |  |

### Write attribute (IAS\_CIE\_Address)

### Write attribute response

### Read attribute (Zone Status)

### Read attribute response

### Zone Status Change

### IAS\_CIE\_Address attribute

### Zone Status attribute

### Zone Status attribute

# Generic devices

In this chapter will be described operations not strictly related to a specific device type that are relevant for a consistent usage experience/service implementation.

## Device Binding and Attribute Reporting

# History of changes