Atmel

APPLICATION NOTE

Atmel AT02597: ZigBee PRO Packet Analysis with Sniffer

Atmel MCU Wireless

Features

- ZigBee[®] Protocol Analysis with Industry-standard sniffing tools such as Luxoft's BitCatcher, Wireshark and Perytons
- Provides instructions and examples on using the sniffer to analyze and debug network behavior

Description

Monitoring Wireless network activity in ZigBee networks is essential in understanding the behavior and traffic patterns of the network. Network Protocol analyzers are powerful and flexible tools that aid debugging during ZigBee-based application development.

This Application Note describes how to configure and use supported sniffing tools along with Atmel[®] MCU-based sniffer hardware platforms.

Table of Contents

1.	Overview	3
	1.1 The Wireshark Capture Interface from Atmel Gallery	
	1.1.1 Setup Information	
	1.2 BitCatcher	
	1.2.1 Setup Information	5
		0
2.	Sniffer Capture Session Setup	7
	2.1 Wireshark Capture Interface	7
	2.2 BitCatcher	8
	2.3 Perytons Network Analyser	
3.	Configuring Sniffer Preferences	
	3.1 Wireshark Capture Interface	
	3.2 BitCatcher	14
	3.3 Perytons Network Analyzer	15
4	Analyzing Data Traffic in ZigBee PRO Networks	17
••	4.1 ZigBee Frame Format Overview	17
	4.2 MAC Association	
	4.3 Network Rejoin with and without PDS	
	4.4 Self-Leave and Parent-Induced Leave	
	4.5 Network Link Status Frame	
	4.6 Multicast	
	4.7 Fragmentation	
	4.8 Service Discovery	
	4.9 I unneling in Secure Networks	26
5.	Analyzing Data Traffic in ZigBee PRO Public Profile N	letworks28
	5.1 ZigBee Home Automation	
	5.1.1 Service Discovery	
	5.1.2 Attribute Reporting	
	5.1.4 Security	29 29
	5.2 ZigBee Light Link	
	5.2.1 Touchlinking	
	5.2.2 Cluster Commands	
	5.2.3 Security	
	5.5 Zigbee Smart Energy	
6.	Example Application Scenarios	31
	6.1 PAN Same Channel Co-Existence	
	6.2 End-to-End Establishment of Application Link Key	
	6.3 ZLL – ZHA Interoperability	
7.	References	
8	Revision History	34
υ.		

		•
	tm	nel
-		

1. Overview

In ZigBee networking, a sniffing tool is important during development and testing for the capture and analysis of frames exchanged in the network. It is more significant in networks that have ZigBee products from different vendors to test and verify that they inter-operate with one another.

The sniffing tool shall be capable of real-time capture of frame formats supported by the ZigBee protocol and the IEEE[®] 802.15.4 standard. The sniffer shall also provide parsed information of different fields and sub-fields of the frame, which shall aid the user in quick analysis.

A visual view of the network topology, time stamping, multi-channel capture and saving of the capture files are some of the features of sniffers that the user can benefit from, to gain a complete picture of the wireless environment that is being monitored.

The Wireless network sniffer environment is setup by running one of the network protocol analyzers in Figure 1-1 on the PC.

Figure 1-1. Supported ZigBee Network Protocol Analyzers.



BitCatcher

Wireshark

Perytons

To start capturing frames on an IEEE 802.15.4 channel, one should have a sniffer hardware tool running a sniffer firmware, plugged into the PC.

Figure 1-2. Supported Sniffer Hardware Platforms.





Atmel RZUSBSTICK

Dresden Elektronik deRFusb23E06



deRFusb13E06

The RZUSBSTICK and the deRFusb23E06 [10] can be used to sniff IEEE 802.15.4 2.4GHz operating channels.

The deRFusb13E06 [10] can be used as an IEEE 802.15.4 sub-GHz sniffer.

1.1 The Wireshark Capture Interface from Atmel Gallery

A simple Capture Interface that can be used to create capture sessions for IEEE 802.15.4 2.4GHz channels is available for download from Atmel Gallery.

Apart from the RZUSBSTICK, this interface supports the RZ600-RF231 radio board [2] as a sniffer hardware platform.



1.1.1 Setup Information

To setup the capture session using Wireshark, download the Atmel Wireshark Interface from Atmel Gallery [7]. Wireshark version *Wireshark-win32-1.8.0* or lesser shall be installed on the PC [8].

Figure 1-3.	Atmel Wireshark Interface in Atmel Gallery.
-------------	---



- The package is installed in directory \Atmel\AtmelWiresharkFirmware
- Follow the instructions in the Readme_Important_Atmel_Wireshark_Sniffer_Interface.txt to complete the setup procedure
- The sniffer firmware to be flashed on the respective hardware platforms, RZUSBSTICK and RZ600 are provided as images (Wireshark_Sniffer_Raven.hex and Wireshark_Sniffer_RZ600.elf) in the package
- Refer to [4] and [2] for programming instructions

Caution: For the RZUSBTICK, the fuse settings shall be:

Extended: 0xFB, High: 0x91, Low: 0xDE

- This version of the Wireshark capture interface (v1.0) does not support sub-GHz sniffer tools
- The USB drivers available with the package support Windows[®] XP/2000/7 and are automatically loaded to Windows by the installer.

Chapter 2 provides further information on setting up a capture session for the Wireshark Interface.



1.2 BitCatcher

Luxoft BitCatcher ZigBee Network Analyze tool is a simple and flexible protocol analyzer that can be used to analyze standard IEEE 802.15.4 and ZigBee frames using Atmel hardware sniffing tools in the sub-GHz and 2.4GHz frequency bands.

1.2.1 Setup Information

The sniffer PC tool can be downloaded from [3].

The package also comes with the firmware files and drivers for the supported sniffer hardware:

- Atmel AVR[®] RZUSBSTICK
- Dresden Elektronik deRFusb-23E00 (2.4GHz) and deRFusb-13E00 (Sub-GHz)

Once installed, sniffer device can be configured for use with the sniffer PC tool as below.

- Plug in the sniffer hardware (e.g.: RZUSBSTICK) to a free USB port in the PC
- Flash the sniffer firmware image to the MCU on the sniffer hardware, if not pre-programmed

Caution: For the RZUSBTICK, the fuse settings shall be:

Extended: 0xFB, High: 0x91, Low: 0xDE

- The firmware images for the RZUSBSTICK and the deRFUSB is available as .hex files in the directory \embedded inside the BitCatcher package
- Un-plug and plug the device into the USB port once again. Install the driver provided in the sniffer package when prompted for
- The sniffer drivers are available in the directory \driver inside the BitCatcher package
- After driver installation, the device will be listed in the Windows Device Manager as shown in Figure 1-4

Figure 1-4. Sniffer Devices Listing in Windows Device Manager.



1.3 Perytons Network Analyzer

The Perytons Analyzer is a powerful sniffing tool with support for Atmel sniffer hardware platforms for both 2.4GHz and sub-GHz IEEE 802.15.4 bands. This application note gives only a brief overview of its features and analysis of data captures with the Perytons Network Analyzer. Further information can be found at [9].

2. Sniffer Capture Session Setup

This chapter shows the user how to setup a sniffer capture session after the installation of the hardware driver and the sniffer tool in the PC has been done successfully.

2.1 Wireshark Capture Interface

- Open Wireshark_Sniffer_Interface.exe from \Program Files\Atmel\AtmelWiresharkFirmware
- From the Sniffer Port column, select the Sniffer port number and click Open as shown in Figure 2-1

Figure 2-1. COM Port Selection.

🔨 Atmel Wireshark Interface Sniffer 🛛 🛛 🔀						
Atmel	Wireshark Sniffer Interface					
Sniffer Port	COM40 V Open					
Channel	11 🗸 Set					
Start Sniffer Status: Open Sniffer Port	Pause Stop Resume					

- As the Sniffer Capture interface supports 2.4GHz channels, the user can select a channel from Channel 11 to Channel 26
- After the channel is set, click Start to begin the capture session as shown in Figure 2-2. A Wireshark instance
 will be invoked automatically



🖭 Atmel Wireshark Interface Sniffer 🛛 🛛 🔀							
	Atmel	Wireshark Sniff	er Interface				
	Sniffer Port	COM40 🗸	Open				
	Channel	17 👻	Set				
		Pause					
	Start		Stop				
	Sniffer Status:	Resume					
	Click Start to capture	in Wireshark					

• After the capture session is complete, the user can save the capture file for further analysis

2.2 BitCatcher

• Open BitCatcher and add a new sniffer device as shown in Figure 2-3 with the port settings in Figure 2-4

Figure 2-3. Adding a New Sniffer Device.

🤪 BitCatcher			
File Window Help			
Add/Remove/Edit device 🔽	Channel pages	V Channels	~ ►
Packet view			
Clone	🛓 Device dialog		
	RZ541 Sniffer	ection	
	RZRaven Sniffer		
	DE Sniffer		
		(OK
Packets log Folding packets log		l	

• It is possible to edit or remove the connection using the same menu



Figure 2-4. Port Settings.

실 Create device 🛛 🔀						
Port COM17	~					
Bit rate	38400	~				
Data bits	8	~				
Stop bits	1	$\mathbf{\mathbf{v}}$				
Parity	None	$\mathbf{\sim}$				
HW conrol	None	$\mathbf{\mathbf{v}}$				
Create Cancel						

 Once the device is created, it will appear in the device drop-down menu in BitCatcher and shall be selected as shown below. After device selection, the connect/Disconnect button shall be pressed to begin or end the usage of the sniffer hardware with BitCatcher

Figure 2-5. Device Selection.

💫 BitCatcher							
File Window Help							
RZRaven Sniffer COM13	Channel pages	✓ Channels	~				
Packet view Clone	Connect/Disconnect						

- After the device connection with BitCatcher is established, the user can configure the Channel and Channel pages supported. Standard IEEE 802.15.4 channels are supported in the 2.4GHz, 868 and 915MHz frequency bands
- Supported Channels in the 2.4GHz bands are from Channel 11 to channel 26 as per formula,
 - Fc = 2405 + 5 (k 11) in [MHz], for k = 11, 12... 26 where k is the channel number.
- Channel 0 is for 868MHz operation and Channels 1 to 10 for 915MHz band. IEEE 802.15.4 -2006 specified channel pages 0, 2 and 5 are supported
- The Data capture session can be started or stopped using the Play Button in BitCatcher

Figure 2-6. Starting the Data Capture Session.

File Window Help				
RZRaven Sniffer COM13 🛛 🗸 🕅 Channel pages	Y	Channel - 0x0B (11)	~	All packets
Packet view		Channel - 0x0B (11) Channel - 0x0C (12)	^	Start/Stop captur
Clone		Channel - 0x0D (13) Channel - 0x0E (14)	=	Edit
		Channel - 0x0F (15)		
		Channel - 0x10 (16)		
		Channel - 0x11 (17)	_	
		Channel - 0x12 (18)	~	

• BitCatcher allows users to save the sniffer capture data in the form of .dcf files and it is also possible to open sniffer logs created using other sniffing tools, if the logs are in the .dcf format

File->Open log/Save log options serve these purposes.

2.3 Perytons Network Analyzer

The Data capture session in Perytons Network Analyzer can be started using the Data Capture button as shown in Figure 2-7.

Figure 2-7. Starting the Data Capture Session.

🎽 Peryton-M4 : NONAME : «No File» : ZigBeeProtocol								
File	Message View	Network View	Time View	Devices View	Tools	Windows	Help	
1	🖻 🛛 🖷 🥚		🔒 🔯 🚰	🔩 🍼 📀	• B	🔁 🍘 🤹	🥜 🗂 💵 🔳 🕻 🐉 🔞)

Before capture is started, it shall be ensured that the hardware is detected by the sniffer tool and the settings are as per Figure 2-8.



Capture 802.15.4/ZigBee/RF4CE/6LoWPAN Protocol						
General Simulate Capture From File Automatically Update Capture Filename						
Data File: C:\capture.anl						
Device File:						
Floor Plan:						
Description:						
Active Protocol: ZigbeePro						
Dongle Type: Atmel 2.4Ghz Refresh Dongles						
AII 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26						
Search For Activity Active Sync V Diversity						
DataCapture						
Max Messages: 500 🗘 🔽 Continuous Message#:						
Max Time [sec]: 70000 💠 Elapsed Time:						
Start (Real-Time) Save & Continue Pause Stop						
Real-Time Capture						
Off-Line Capture						
Ready						

Figure 2-8. Configuring the Capture Parameters.

The user can save the capture file using the Save & Continue option shown in Figure 2-8.

3. Configuring Sniffer Preferences

For easier viewing and analysis, sniffers GUIs provide multiple filtering options. With the appropriate settings, a complete snapshot of the wireless network can be obtained. This chapter provides information on configuring such preferences in the sniffer GUI.

3.1 Wireshark Capture Interface

- **Protocols**: Wireshark automatically identifies the protocol in use, as all supported protocols are enabled by default, as can be seen in menu option Analyze -> Enabled Protocols. The user can enable or disable protocols as per requirement using this option
- Security: It is possible to monitor encrypted ZigBee network data by entering the NWK security key used in the network. Use menu option Edit -> Preferences -> Protocols -> ZigBee NWK as shown in Figure 3-1

Figure 3-1. Security Preferences in Wireshark

📶 Wireshark: Preferences - Profile: Default 📃 🗖 🔀						
UNISTIM	ZigBee Network Layer					
USB	Security Level:	AES-128 Encryption, 32-bit Integrity Protection	×			
USBCCID	Pre-configured Keys:	Edit				
VCDU						
VLAN						
VNC						
VSS-Monitoring						
Vuze-DHT						
WBXML						
WiMax (wmx)						
WIMAX ASN CP						
WINS-Replication						
WOW						
X.25						
X11						
X2AP						
XMCP						
XML						
XMPP						
XOT						
YAMI						
YMSG						
ZEP						
ZigBee NWK 😽 😽						
Help		<u>o</u> k	Apply Cancel			

The security level shall be set as per the ZigBee specification, Table 3-1 is taken from the ZigBee specification document and lists the supported security levels.

Security level identifier	Security level sub- field (Table 4.24)	Security attributes	Data encryption	Frame integrity (length M of MIC, in number of octets)
0x00	ʻ000'	None	OFF	NO (M = 0)
0x01	'001'	MIC-32	OFF	YES (M = 4)
0x02	'010'	MIC-64	OFF	YES = (M = 8)
0x03	'011'	MIC-128	OFF	YES (M = 16)
0x04	'100'	ENC	ON	NO (M = 0)
0x05	'101'	ENC-MIC-32	ON	YES (M = 4)
0x06	'110'	ENC-MIC-64	ON	YES = (M = 8)
0x07	'111'	ENC-MIC-128	ON	YES (M = 16)

Table 3-1. Security Levels – NWK and APS – ZigBee Specification.

It is possible to add multiple keys, edit or remove existing keys as shown in Figure 3-2.

Example: In a ZigBee network using security in the APS layer, a device joining the network establishes a link key with the Trust Center. In order to view all APS transactions for this link such as the APS Transport Key command, the Trust Center link key needs to be added to the sniffer preferences.

Figure 3-2. Security Key Entries.

🗖 Pre-configured K	eys - Profile: Default			
			6 C	
	AA:AA:AA:AA:AA:AA:AA:AA:BB:BB:BB:BB:BB:B	Normal	r Labei Trust Center Link Key	
Up D	AA:AA:AA:AA:AA:AA:AA:AA:BB:BB:BB:BB:BB:B	Normal	key1	
Down	AA:AA:AA:AA:AA:AA:AA:AA:BB:BB:BB:BB:BB:B	Normal	Key2	
New				
Edit				
Copu				
Delete				
Refresh				
_lear				
				Cancel

- User Interface Options: It is possible to customize viewing options in the Wireshark interface
 - Arrange the layout of the panels from Edit -> Preferences -> Layout
 - Add Columns to the packet display pane (e.g.: HW src addr) from Edit -> Preferences -> Columns
 - Colorize frame formats (e.g.: NWK Link Status Frames) from View -> Coloring Rules
 - Apply Filters to display frames based on chosen fields in a frame by right-clicking on the field and using
 option Apply as Filter



Figure 3-3. Wireshark Capture Screen Layout.

No	HW/Src Addr	HW/ Dect Addr	NWK Src Addr	Protocol	Info		Frame 11: 56 but as on wire (448 bits) 54 but as cantu
140.	1	Broadcast	NWK DIC AUU	TEEE 802 15 4	Beacon Request	8.	Frame II. Jo bytes on white (448 bits), J4 bytes caption
	2.0×0000	Broadcast	0×0000	710800	Link Status	II	TEEE 802.13.4 Data, DSt: 0X9TT7, SFC: 0X0000
	2 0,0000	BIUAUCASC	0,0000	Ziybee	LINK SLALUS		ZigBee Network Layer Data, Dst: 0x9ff/, Src: 0x0000
	3	Broadcast		IEEE 802.15.4	Beacon Request	Ξ.	ZigBee Application Support Layer Command
	4 0x0000			ZigBee	Beacon, Src: 0x0000, EPID: aa:aa:aa		■ Frame Control Field: Command (0x01)
	5 00:00:00:01:00:00:00:00	0×0000		IEEE 802.15.4	Association Request		Counter: 210
	6			IEEE 802.15.4	Ack		Command Frame: Transport Key
	7 00:00:00:01:00:00:00:00	0x0000		IEEE 802.15.4	Data Request		Command Identifier: Transport Key (0x05)
	8			IEEE 802.15.4	Ack		Key Type: Standard Network Key (0x01)
	9 aa:aa:aa:aa:aa:aa:aa:aa	00:00:00:01:00:00:	0	IEEE 802.15.4	Association Response, PAN: 0x1aaa A		Kev: aaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbb
	LO			IEEE 802.15.4	Ack		Sequence Number: 0
	L1 0×0000	0x9ff7	0×0000	ZigBee	Transport Key		Extended Destination: 00:00:00 01:00:00:00:00 (0
	L2			IEEE 802.15.4	Ack		Extended Source: aa:aa:aaaa:aa:aaaa:aa (aa:aa:aa
	L3 0x9ff7	Broadcast	0x9ff7	ZigBee ZDP	Device Announcement, Device: 00:00:		
	L4 0×0000	Broadcast	0x9ff7	ZigBee ZDP	Device Announcement, Device: 00:00:		
	L5 0x0000	Broadcast	0x0000	ZigBee	Link Status		
	L6 0×9ff7	Broadcast	0x9ff7	ZigBee	Link Status	000	00 61 88 5d aa 1a f7 9f 00 00 08 00 f7 9f 00 00 01
	L7 0x0000	Broadcast	0x0000	ZigBee	Link Status	001	10 46 01 d2 05 01 aa aa aa aa aa aa aa aa bb bb bb
	L8 0×9ff7	Broadcast	0x9ff7	ZigBee	Link Status	002	20 bb bb bb bb bb 00 00 00 00 00 01 00 00 00 aa aa
	L9 0×0000	Broadcast	0×0000	ZigBee	Link Status	00:	su aa aa aa aa aa

3.2 BitCatcher

- **Protocols**: By default, BitCatcher is automatically able to parse the following protocol packets:
 - IEEE 802.15.4 MAC
 - ZigBee PRO (Network, ZDO and APS layers)
 - ZigBee Public Profiles such as ZigBee Smart Energy, Home Automation, and Light Link

It is not possible to explicitly disable any of the above protocols in BitCatcher, although it is possible to enable or disable packet views of various ZigBee PRO layers (NWK, APS and ZCL).

This configuration can be done using menu option Window -> Sniffer -> Preferences ->Layers

• Security: It is possible to monitor encrypted ZigBee network data by entering the NWK security key used in the network. Use menu option Window -> Sniffer -> Preferences -> Core security as shown in Figure 3-4

Figure 3-4. Security Preferences in BitCatcher.

Preferences		
Core security	Security level 5	
🕂 🔶 Filters	Кеу	1.1
 Layers 	CB:DE:C9:EB:E4:EF:7F:54:E1:62:FF:92:C0:28:E0:71	
····· • ZLL security	C7:71:C7:BF:6C:C6:1B:7A:6A:1C:00:72:0B:93:13:C7	
	B3:B1:07:ED:54:49:CD:39:ED:38:ED:C0:D4:74:CB:38	
	BF: 5B: F6: 23: 7D: 72: 3B: 1E: 27: 00: 54: 48: AA: 72: DA: E4	
	97:18:E5:58:FD:5B:29:33:70:D7:EA:F2:17:11:A6:3D	
	17:CC:AD:31:A2:18:8E:0E:32:B0:42:00:F4:DA:23:D3	
	C5:F7:92:9D:3F:E2:5B:7D:CC:B9:D4:0C:25:0F:31:0D	
	67:1A:E1:D3:5F:9A:75:27:9F:15:EA:F1:F3:CB:72:28	
	F1:CF:48:F5:CE:BF:30:AA:30:BF:CF:65:D9:39:0D:2C	
	CD:17:52:F3:CC:49:75:BE:1A:34:1B:1D:B5:D9:5A:FF	
	OF:9B:F9:EB:90:78:5B:CF:BE:69:E7:C6:ED:80:48:DE	
	8E:A4:2D:46:F9:09:ED:86:37:01:70:FF:1E:B9:58:95	
	4D:99:E7:5C:19:E7:BD:6F:D1:78:54:4E:96:9F:E0:14	2
	18:54:10:2E:D1:EC:C8:14:56:E2:0C:C2:20:45:70:AD	2
	B7:47:45:5B:59:58:5B:05:17:0B:D0:4B:94:2D:5B:23	2
	D9:80:27:01:01:33:DE:F9:BF:64:F1:B8:83:11:03:24	
	FC:BE:5A:71:EE:5A:DD:E0:CC:0F:17:69:BA:5F:C0:3A	
	9D:3D:27:6A:41:BD:16:C7:52:A5:32:A5:9D:50:04:5A	
	A0:57:A3:3A:63:3E:94:B6:9E:2A:64:68:59:59:D3:71	
	2C:38:74:03:D4:1E:E0:B5:8C:60:C1:2A:F9:E9:F7:3A	
	NOTE: Key type is detected automatically	d Remove
	Apply Drop changes	Ok Exit

The security level shall be set as per the ZigBee specification, Table 3-1 is taken from the ZigBee specification document and lists the supported security levels.

ZigBee Light Link (ZLL) security keys used for inter-PAN data exchange can also be configured in this sniffer using menu option Window -> Sniffer -> Preferences -> ZLL security.

It is possible to add multiple keys, edit or remove existing keys as shown in Figure 3-4.

If the key is not added to sniffer, the packets shall not be decrypted and shall be displayed in a turquoise blue color font.

- **User Interface Options**: It is possible to customize viewing options in BitCatcher
 - The default layout of the sniffer provides a snapshot of the packet log, a timeline view as well as a packet dissector called the Packet view. It is possible to drag and re-arrange the views as per need and also close views
 - The menu option Window -> Show view can be used to show a view closed by the user
 - Add Columns to the packets log panel (e.g.: HW src addr) by right-clicking on any column in the panel and selecting the option Manage Columns
 - Packets are colorized depending on the layer at which they are formed and sent out (NWK, APS, and MAC).

Apply Filters to display frames based on chosen fields in a frame by using menu option Window -> Preferences -> Filters

3.3 Perytons Network Analyzer

- **Protocols**: Perytons supports the following protocols:
 - IEEE 802.15.4 MAC
 - ZigBee PRO (Network, ZDO and APS layers)
 - ZigBee Public Profiles such as ZigBee Smart Energy, Home Automation, and Light Link

The active protocol can be set when the capture session is configured as shown in Figure 2-8.

• Security: It is possible to monitor encrypted ZigBee network data by entering the NWK security key used in the network. Use menu option Window -> Sniffer -> Preferences -> Core security as shown in Figure 3-5

Figure 3-5. Security Preferences in Perytons.

Keys Management	(1 ke	ys)						$\overline{\mathbf{X}}$			
🗔 🗔 🍃 🚺	20		-	_	_	-	_				
Default Security Lev	el:	Default (according to key type)	~	🗹 Try	All Known Ke	ys					
MAC2003 Security S	uite:	7, AES-CBC-MAC-32	~	🗹 Us	e Keys From F	ile	Try All Fi	ile Keys			
		1.000.000					Key Editin	g			X
NasterKey	2N 1.G	516967426565416C6C69616163653039		vicel	Dearces	F	Type: [Key: PAN ID: [Level: [Last Time [LinkKey CO C1 C2 C3 C4 C C8 C9 CA CB CC C OxDEAD Device1: Default (taken from key pa Jsed: Never	:5 C6 C7 D CE CF 0x000 rameters)	AAAAARC ÈÉÉÈIÍÍÍ Device2:	0x1234 Added In File: No
Add	dodify	Delete									OK Cancel



The security level shall be set as per the ZigBee specification, Table 3-1 is taken from the ZigBee specification document and lists the supported security levels.

- User Interface Options: It is possible to customize viewing options in Perytons
 - The tool provides different viewing options such as the Time view, Network view, Message view, Statistics, and Message Tree view. The user can enable or disable views using menu option Windows
 - Add Columns to the packets log panel (e.g.: HW src addr) by right-clicking on any column in the panel and selecting the option Manage Columns
 - Packets are colorized as per settings in Tools -> Preferences
 - Apply Filters to display frames based on chosen fields in a frame by using menu option Window -> Preferences -> Filters

Figure 3-6. Perytons Example Capture Window.

Peryton-M4 : NONAME : capture.anl : ZigbeePro			_ 🗆 🔀
File Message View Network View Time View Devices View Tools Windows Help			
i D 🗠 🖟 👘 🌘 🖬 🔲 👁 🔒 🗟 🖓 者 🎸 🔇 🎚 🎘 🖏 🖓 🌾 🖉 🗉			
Ime View	Network	(Yiew	- ×
Grouping: Device - GoTo: 0 🕞 😫 🔎 🛄 🖑 🔞 🝸 - 🖈 🗿 🗔 🚔 🔞	Network Index:	- All - 🔽 🖉 🖓	• 🔔 🕹 📮
Unknown 2 2 1 N N C N N Z Z MA N	0000000	000000000000000000000000000000000000000	
0.009< 4.9685			~
	<)		>
Message View	- ↓ ×	Message Tree View	ą×
Image: Second Log I	Dev Mac		
Msg Into Msg # StartTime PCstartTime TimeDiff Len SS Channel Layer Type RM AckTo AppStat Status 16 16 5001368 10/7/13 17:37:00.008 1825 5 172 17 Mac Ack 15 OK OK	Dev Mac 1		
Msg Info Msg # StartTime PCStartTime TimeDiff Len SS Channel Layer Type RM FrwOf AppStat 17 17 5058006 10/7/13 17:37:00.070 56638 54 172 17 ZDP MatchDescReq 12 OK	t Status OK	Hex Data	
Msg Info Dev 1 Mac Data Header FType Sec Pen AckR CPan R1 DMode Ver SMode Seq DPanId DstAdd SrcAdd FFFF 18 10 10 50 0 N N N Y 0 5 03 5 104 11A0 FFFF 0000 Msg 1nfo Dev Mac Data Header FType Sec Pen AckR CPan R1 DMode Ver SMode Seq DPanId DstAdd SrcAdd FFF 19 10 2 Mac Data Header FType Sec Pen AckR CPan R1 DMode Ver SMode Seq DPanId DstAddd SrcAdd FFF 19 10 2 30 5 N N N N N 0 S 03 5 102 11A0 FFFF F232	FCS	Message Data: 0000: 61 88 69 A0 11 00 00 9 0010: 28 28 07 00 00 08 7 0020: 2B A0 A7 98 01 40 A9 9 0030: 50 09 94 98 2C 85 51 F 0040: 40 C7	9 60 48 7 66 55 4 CE 90 2 DC 61
	>	0010:	 0 01 00 V 07 70 V

4. Analyzing Data Traffic in ZigBee PRO Networks

This chapter provides examples of common interaction in ZigBee PRO networks that shall aid the user to look closely into various fields of the frame and is not aimed at covering all scenarios that fall under ZigBee specification. Explaining the meaning of all the fields used in a ZigBee frame is outside the scope of this application note and shall be looked up in the ZigBee specification [6].

All Configuration Server (CS) parameters mentioned in this Application Note can be set from application configuration.h or at run-time using CS_WriteParameter API. The description of the CS parameters is also present in respective API_Reference.chm available in the \Documentation folder in the BitCloud[®] SDK. Information on when the parameter shall be set (at compile time only, at run-time before network start or at any time) along with parameter persistence is available in this file.

4.1 ZigBee Frame Format Overview

Figure 4-1 shows the skeletal overview of the ZigBee frame format.



Figure 4-1. Frame Format.

The APS and NWK layer security headers and footers are also shown in the Figure 4-1. ZigBee uses a non-beacon enabled MAC format with no security in the MAC layer.

4.2 MAC Association

Every node in a ZigBee network has its own unique 64-bit IEEE/MAC address. When a node joins the network for the first time, a MAC association procedure is performed, following which it obtains a 16-bit network (short) address from the parent. This short address is used for further communication in the network, to reduce frame overhead.

BitCloud provides a configuration parameter in application called CS_UID which can be used to set the value of the 64bit MAC address of the node during compilation. The following conditions prevail:

- Setting CS_UID for testing: Any random 64-bit value can be set at compile time in application configuration.h or at run-time before calling ZDO_StartNetworkReq()
- Setting CS_UID during production: Commercial usage of ZigBee products require purchase of a block of IEEE/MAC addresses from IEEE. This can be programmed into an external EEPROM specifically used for this purpose. In this case, CS_UID can be set to zero during compile-time, BitCloud will attempt to read the value from the external EEPROM in this case using API, HAL_ReadUid()



When a node starts up, it does network discovery by performing an active scan over the channels specified in the CS_CHANNEL_MASK parameter in BitCloud. A beacon request is sent as seen in packet #4 in Figure 4-2.

Upon reception of a beacon request frame, routers and coordinators already present in network automatically respond with a beacon frame. The joining node filters a potential parent based on the settings in the beacon packet received.

4	11:23:42.617	MAC: MacBeaconReg		0×ffff		1
5	11:23:42.619	Beacon	0×0000			
6	11:23:43.123	MAC: Mac Association Re	0x0000000000000055	0x0000		
7	11:23:43.123	Ack				
8	11:23:43.620	MAC: Mac Data Request	0×000000000000055	0x0000		
9	11:23:43.620	Ack				
10	11:23:43.623	MAC: Mac Association Re	0×000000000000000000000000000000000000	0x000000000000055		
11	11:23:43.623	Ack				
12	11:23:43.749	ZDO: DeviceAnnce	0x5f53	0×ffff	0x5f53	0xfffd
13	11:23:43.786	ZDO: DeviceAnnce	0x0000	0×ffff	0x5f53	0xfffd
14	11:23:45.227	NWK: Link Status	0×0000	0×ffff	0×0000	0xfffc
15	11:23:45.628	NWK: Link Status	0×5f53	0×ffff	0x5f53	0×fffc

Figure 4-2. Node Joins Network Via MAC Association Using IEEE Address 0x55ULL.

The beacon from the coordinator/router contains the Association Permit subfield, it is set to TRUE if the device is accepting association to the PAN. A joining node cannot associate to the device if this sub-field is FALSE.

CS_PERMIT_DURATION controls the joining of devices into the network by setting the permit duration.





Example: Figure 4-4 shows parsed beacon payload that contains information based on which the joining node chooses a potential parent.

The beacon payload provides information on the ZigBee stack profile used in the network (ZigBee PRO = 0x02), network protocol version (*nwkcProtocolVersion*, 0x02 as per ZigBee PRO specification). The router capacity, end-device capacity and device depth limits the acceptance of children by a parent node.

Figure 4-4. Beacon Payload.

i≘beacon
⊕superframe : 0x8FFF
∰⊸GTS
⊕⊷pending
⊜⊷beaconPayload
protocolId : 0x00
⊨-prop : 0x8C22
stackProfile : 0x02
nwkProtoVer : 0x02
reserved : 0x00
routerCapacity : 0x01
deviceDepth : 0x01
edCapacity : 0x01
extPanId : AA-AA-AA-AA-AA-AA-AA-AA
txOffset : 0xFFFFFF
updateId : 0x00

Table 4-1. Config. Server Parameters Affecting Beacon Payload.

Capability info sub-field	Config. server parameter		
routerCapacity	CS_MAX_CHILDREN_ROUTER_AMOUNT		
edCapacity	CS_MAX_CHILDREN_AMOUNT		

A joining device indicates its capability information in the MAC association request it sends to its potential parent as shown in Figure 4-5 and Table 4-2.

Figure 4-5. Capability Information in a MAC Association Request.

```
macCommandType : Mac Association Request (0x01)
-capabilityInfo : 0x8E
---altPanCoord : 0x00
---deviceType : 0x01
---powerSource : 0x01
--rxOnWhenIdle : 0x01
--reserved : 0x00
---securityCap : 0x00
---allocateAddress : 0x01
```

Table 4-2. Config. Server Parameters Affecting Capability Info.

Capability info sub-field	Config. server parameter		
deviceType	CS_DEVICE_TYPE		
rxOnWhenIdle	CS_RX_ON_WHEN_IDLE		



Without a proper setting of the CS_UID parameter, a node tries to repeatedly join the network CS_ZDO_JOIN_ATTEMPTS times unsuccessfully.

MAC: MacBeaconReg		Oxffff
MAC: MacBeaconReg		Oxffff
Beacon	0×0000	
NWK: Link Status	0×0000	Oxffff
MAC: Mac Association Request	0×000000000000000	0×0000
Ack		
MAC: Mac Data Request	0×000000000000000	0×0000
Ack		
MAC: Mac Association Request	0×000000000000000	0x0000
Ack		
MAC: Mac Data Request	0×000000000000000	0x0000
Ack		
MAC: MacBeaconReg		Oxffff
Beacon	0×0000	
MAC: Mac Association Request	0×000000000000000	0×0000
Ack		
MAC: Mac Data Request	0×000000000000000	0×0000
Ack		
MAC: Mac Association Request	0×000000000000000	0×0000
-		

Figure 4-6. Node Tries to Join Network Without Proper Setting of IEEE Address.

4.3 Network Rejoin with and without PDS

There are multiple join options that can be configured using BitCloud as mentioned in [1].

Network rejoin method is used when a node wishes to join a network with knowledge of networking parameters such as the network's extended PANID or application-assigned short address, operating channel or others.

Figure 4-7 shows examples of several successful network rejoin procedures.

Figure 4-7. Node Joins Network Via Rejoin Procedure with PDS Not Used in Application.

Frame	MAC Src	MAC Dst	NWK Src	NWK Dst
MAC: MacBeaconReg		0×ffff		
MAC: MacBeaconReg		0×ffff		
Beacon	0×0000			
NWK: Link Status	0×0000	O×ffff	0×0000	0×fffc
NWK: Rejoin Request Ack	0×0001	0×0000	0×0001	0×0000
NWK: Rejoin Response Ack	0×0000	0×0001	0×0000	0×0001
ZDO: DeviceAnnce	0×0001	0×ffff	0×0001	0×fffd
ZDO: DeviceAnnce	0×0000	O×ffff	0×0001	0×fffd
NWK: Link Status	0×0000	O×ffff	0×0000	0×fffc
NWK: Link Status	0×0001	O×ffff	0×0001	0xfffc
MAC: MacBeaconReg		O×ffff		
Beacon	0×0000			
NWK: Rejoin Request	0×0001	0×0000	0×0001	0×0000
Ack				
NWK: Rejoin Response	0×0000	0×0001	0×0000	0×0001
Ack				
ZDO: DeviceAnnce	0×0001	0×ffff	0×0001	0×fffd
ZDO: DeviceAnnce	0×0000	0×ffff	0×0001	0×fffd
NWK: Link Status	0×0001	0×ffff	0×0001	0×fffc
NWK: Link Status	0×0000	0×ffff	0×0000	0×fffc

The Persist Data Server (PDS) Component provides useful API to store networking parameters to persistent memory to retain them over device resets. See section PDS in [1]. With PDS APIs used in application, the node can be configured to come back into the network without performing over-the-air rejoin procedure.

In Figure 4-8, the node with short address 0x0001 joins network via rejoin from packet #3.

Just before packet #13, this node is reset and it is seen that the device does not do further rejoin to come into the network.

The node reset behavior and further restoration of configuration server parameters with PDS can be verified by looking at the values of the sequence numbers and counter values in various layers. These parameters are among the config. server parameters and hence are restored on rejoin with PDS implemented. Without PDS implementation, the stack resets these values to default values on startup.

Packet N	Frame	MAC Src	MAC Dst	NWK Src	NWK Dst
o	MAC: MacBeaconReg		0×ffff		
1	NWK: Link Status	0×0000	O×ffff	0×0000	0×fffc
2	NWK: Link Status	0×0000	O×ffff	0×0000	0xfffc
з	MAC: MacBeaconReg		O×ffff		
4	Beacon	0×0000			
5	NWK: Rejoin Request	0×0001	0×0000	0×0001	0×0000
6	Ack				
7	NWK: Rejoin Response	0×0000	0×0001	0×0000	0×0001
8	Ack				
9	ZDO: DeviceAnnce	0×0001	0×ffff	0×0001	0×fffd
10	ZDO: DeviceAnnce	0×0000	0×ffff	0×0001	0×fffd
11	NWK: Link Status	0×0001	O×ffff	0×0001	0xfffc
12	NWK: Link Status	0×0000	O×ffff	0×0000	0xfffc
13	NWK: Link Status	0×0001	O×ffff	0×0001	0xfffc
14	NWK: Link Status	0×0000	0×ffff	0×0000	0×fffc
15	NWK: Link Status	0×0001	O×ffff	0×0001	0xfffc

Figure 4-8. Node Joins Network Via Rejoin Procedure with PDS Used in Application.

The silent rejoin is very useful when nodes are reset after a battery change, or after a mains-powered network has reset after a power outage. The nodes come back into the network seamlessly and can route packets, with their networking tables restored from Persistent memory to RAM.

Packet #9 shows a device announcement frame. This broadcast frame is sent by a device after it has properly authenticated into the network.

Devices that receive this frame update their network address map tables with the device information obtained in the device annce frame.

4.4 Self-Leave and Parent-Induced Leave

ZDP Requests are managed by the ZDO layer and can be used for various network control scenarios as detailed in [1].

Network leave can be processed using ZDP requests when a device needs to leave the network on certain events. Network leave can be self-induced on a node or a node can order another remote node to leave the network.

Figure 4-9. Self-Leave of Node With Short Address 0x0001 and Extended Address 0x02ULL.

Packet view				
Clone 🔄 Edit				
detShovtAdde + 0vEE	ED			
	FD			
srcShortAddr : UXUU	J1			
radius : 0x01				
seqNo : 0x06				
srcExtAddr : 00-00-0	0-00-00-00-00-02			
nwkCommandType : Lea	ve (0x04)			
eave : 0x80				
reserved : 0x00				
rejoin : 0x00				
request : 0x00				
removeChildren : 0x0	01			
• parameters				
	0 00 10 ED EE 01 00 01 06 0	2 00 00 00 00 00 00 04 00		
		2 00 00 00 00 00 00 00 04 80		
Packets log				
Cols All packets	🗸 Create Edit I	Remove 🔽 Auto scroll		
Time	Packet N	Frame	MAC Src	MAC Dst
01:32:38.847	þ	MAC: MacBeaconReg		0×ffff
01:32:39.530	1	MAC: MacBeaconReg		0×ffff
01:32:39.545	2	Beacon	0×0000	
01:32:40.041	3	NWK: Rejoin Request	0×0001	0×0000
01:32:40.041	4	Ack		
01:32:40.046	5	NWK: Rejoin Response	0×0000	0×0001
01:32:40.046	6	Ack		
01:32:40.093	7	ZDO: DeviceAnnce	0×0001	O×ffff
01:32:40.178	8	ZDO: DeviceAnnce	0×0000	0×ffff
01:32:40.223	9	NWK: Leave	0×0001	0×ffff

Figure 4-10. Parent Node 0x0000 Sends a ZDP Request Requesting Child 0x0001 to Leave.

NWK: Link Status	0×0001	0×ffff	0×0001	0xfffc
NWK: Link Status	0×0000	0×ffff	0×0000	0xfffc 👘
ZDO: MgmtLeaveReq	0×0000	0×0001	0×0000	0×0001
Ack				
ZDO: MgmtLeaveRsp	0×0001	0×0000	0×0001	0×0000
Ack				
APS: Ack MgmtLeaveRsp	0×0000	0×0001	0×0000	0×0001
Ack				
NWK: Leave	hx0001	hxuu	hxnnn	uxrrrd

It is possible to configure options such as rejoin, removal of children in the leave request.

4.5 Network Link Status Frame

NWK link status frames are sent by routers and coordinator so that neighboring nodes can maintain information on the link costs required for routing.

Link Status frames are periodically transmitted as one-hop broadcasts. The Link Status list contains the short address and link cost information of all neighboring nodes as shown in Figure 4-12.

Figure 4-11. Header Information in a NWK Link Status Frame.

```
i≜-nwkHeader
   ie frameCtl : 0x1009
     -dstShortAddr : 0xFFFC
     --srcShortAddr : 0x0384
     ---radius : 0x01
     ---seqNo : 0x45
    srcExtAddr : 00-02-12-FF-FF-0B-2C-2E
  -nwkCommandType : Link Status (0x08)
⊖ options : 0x61
     ---entryCount : 0x01
     -firstFrame : 0x01
     -lastFrame : 0x01
    reserved : 0x00
🖮 linkStatusList -
      -[0] : 0x0000
    [1] : 0×01
```

Table 4-3. Config. Server Parameters Relevant to NWK Link Status.

Information	Config. server parameter
Neighbor Table size	CS_NEIB_TABLE_SIZE
Number of missed link status frames to remove an entry from neighbor table	CS_NWK_MAX_LINK_STATUS_FAILURES

4.6 Multicast

Broadcasting a message to a group of nodes involves creating a group table entry for a specified end-point and group ID. Refer to Section *6.4.3 Multicast transmission* in [1] for implementation specifics.

Sniffer log, *multicast.dcf*, shows a multicast transmission from coordinator to Group 1 and end-point 0xF0 in #34.The network destination address is the group address.

Multicasts do not have APS Acknowledgements and are re-broadcast by all routers (group members and nonmembers) until the radius is exhausted. Figure 4-13 shows the non-member radius which is decremented on every nonmember re-broadcast of the frame.



Figure 4-12. Multicast Sub-Field – NWK Header.

```
i i - nwkHeader

    frameCtl : 0x0108

     -dstShortAddr : 0x0001
      -srcShortAddr : 0x0000
     -radius : 0x02
     -seqNo : 0x4D
   ⊜-multicast : 0x49
         -mode : 0x01
         -nonmemberRadius : 0x02
         -maxNonmemberRadius : 0x02
⊡-apsHeader
   ⊕ frameCtl : 0x08
     -dstEp : 0xFF
     --clusterId : 0x001C
     ---profileId : 0x7F01
     -srcEp : 0x01
     -counter : 0xC1
🖻 - data
    [0] : 0x0A
```

In response to #34, group member node 0x31BC responds with APL data on end-point 0xF0 whereas node 0x0BDD is a non-member and so does not respond.

0x0BDD is then added to the group and so responds with APL data on end-point 0xF0 in #59.

4.7 Fragmentation

When APL data packets greater than the maximum size of the APL payload needs to be sent, the stack fragments the entire data into blocks. The fragmentation concept is explained in Section *6.7 Fragmentation* of [1].

Figure 4-13. Fragmentation – Relevant Header Information.

```
--apsHeader
    frameCtl : 0xC0
        ---dstEp : 0x08
        ---clusterId : 0xFC00
        ---profileId : 0xBF0D
        ---srcEp : 0x08
        ---counter : 0x6F
        ---srtHeader
        ---frameCtl : 0x01
        ---blockNumber : 0x02
        ---data
        ---[0] : 0x18
        ---[1] : 0x61
```



Figure 4-14. Fragmentation – Example.

Frame	NWK Src	NWK Dst
APS: Fragmented packet 0x02	0x32fa	0×0000
Ack APS: Eragment 0×01	0×32fa	0~0000
Ack	0X32FG	
APS: Ack 0xFC00	0x0000	0x32fa
Ack		

As per ZigBee specification, the first fragment is sent with the block number as the total number of blocks comprising the entire APL data, as seen in Figure 4-14. The subsequent fragments have block numbers starting from 1 onwards up to maximum transmission window size. The receiving node sends an APS Acknowledgment frame when all blocks in the transmission window have been received.

 Table 4-4.
 Config. Server Parameters Affecting Data Transmission.

Information	Config. server parameter
Single Fragment size	CS_APS_BLOCK_SIZE
Number of fragments /blocks the APL data is split into	CS_APS_MAX_BLOCKS_AMOUNT
Number of blocks to be sent before expecting an APS ACK	CS_APS_MAX_TRANSMISSION_WINDOW_SIZE
Size of a single fragment in fragmented transmission	CS_APS_TX_BLOCK_SIZE
Size of one receive buffer that will hold one fragment of the data frame.	CS_APS_RX_BLOCK_SIZE

4.8 Service Discovery

Service Discovery is the process of collecting information on supported clusters on other devices in the network. Service discovery uses ZDP requests (with MATCH_DESCRIPTOR_CLID) for every cluster ID supported. Service discovery requests can be unicast or broadcast and so the response contains the network address of the responder along with the matched simple descriptor information.

The response contains a match list with the end-points that support the cluster in the request. Service discovery is explained in further detail in Section 5.1.1.

4.9 Tunneling in Secure Networks

Consider a network wherein a node insecurely joins through a router parent (i.e.) the joining node does not know the network key prior to join procedure.

In this case, the APS command used to securely communicate the network key from the Trust Center to the newly joined router is called the APS Tunnel command.

Referring to sniffer log, *Transport_key_tunnel.dcf*, end-device 0x01LL joins router 0x0000000100000000 from packet #26. The parent router sends an APS Update Device command to the Trust Center to inform the Trust Center that a node has joined or left the network.

Figure 4-11 shows the update status of the device, depending on which the Trust Center takes necessary action (i.e.) to send the network or remove the key and associated security counters for the device.

Figure 4-15. Status Field in APS Update Device Command.

Status	Integer	0x00 - 0x07	Indicates the updated status of the device given by the DeviceAddress parameter:
			0x00 = Standard device secured rejoin
			0x01 = Standard device unsecured join
			0x02 = Device left
			0x03 = Standard device unsecured rejoin
			0x04 = High security device secured rejoin
			0x05 = High security device unsecured join
			0x06 = Reserved
			0x07 = High security device unsecured
			rejoin

Trust Center sends the APS Tunnel command frame in #39. The Tunnel command frame will contain the secured frame to be sent to the destination, in its payload.

#43 shows the APS Transport Key command frame sent from the router parent to the newly joined end-device. It includes the key sequence number and the active network key. In case the router has joined with a pre-configured network key, the APS Transport packet would contain key sequence number and the key values as all-zeros.

End-device receives the Transport Key command frame, sets and activates the network key and does a device announcement to the network.



5. Analyzing Data Traffic in ZigBee PRO Public Profile Networks

Networks based on ZigBee PRO Public Profiles such as ZigBee Smart Energy, Home Automation, Light Link employ specific join, security and data exchange schemes. The implementation details of the reference applications can be found in [5].

This chapter provides sniffer data analysis of example sniffer logs taken using [5]. The sniffer logs are provided in a zip package with this application note and this chapter references packet numbers from these logs. The logs are in *.dcf format and can be viewed in BitCatcher.

5.1 ZigBee Home Automation

The Home Automation (HA) reference application in BitCloud Profile Suite implements six HA device types. The sniffer log, *HA_TC_DL_DS.dcf*, demonstrates data exchange between a Trust Center, Dimmable Light and Dimmer switch device.

5.1.1 Service Discovery

The Dimmable Light device joins the network created by the Trust Center from packet #3 and the Dimmer switch from #17. This network employs standard network security with application link keys.

#32 shows a Match Descriptor Request broadcast by the dimmer switch to all devices with *macRxOnWhenIdle* = TRUE (nwkHeader->dstShortAddr = 0xFFFD).

The Match Descriptor Data field consists of the number of clusters and the cluster ID .The request should be issued individually for each server and client cluster. Hence, in #37, the in/client cluster count is 0x01 and cluster ID is the ONOFF_CLUSTER_ID.In #178, the in cluster count is 0x00, the out/server cluster count is 0x01 and the cluster ID is the ONOFF_SWITCH_CONFIGURATION_CLUSTER_ID.

The profile ID is PROFILE_ID_HOME_AUTOMATION defined in *zcl.h* and the cluster related defines in *clusters.h* found in *\ZCL\include directory* in the BitCloud SDK.

The Trust Center does not support the clusters in this request and so simply re-broadcasts the packet (#33 shows this, the apsHeader->counter and nwkHeader sequence number stay same, whereas the nwkHeader->radius is decremented by one on re-broadcast).

The Match descriptor response gives sufficient information about the responder (short address, endpoint and cluster information) for the requester device to do a device discovery (IEEE address determination) and binding.

Every cluster binding creates an entry in the requester's binding table.

Table 5-1. Relevant Config. Server Parameters.

Network information	Config. server parameter
Binding Table	CS_APS_BINDING_TABLE_SIZE
IEEE and short Address pair	CS_ADDRESS_MAP_TABLE_SIZE

5.1.2 Attribute Reporting

The HA reference application sets up automatic reporting of attributes with application configured report period.

In #113, dimmable light reports value of attribute ID, ZCL_ONOFF_CLUSTER_ONOFF_SERVER_ATTRIBUTE_ID (*attributeStatusList*) of the ONOFF_CLUSTER_ID (*apsHeader -> clusterId*).

The ZCL header specifies that the direction of this frame (server to client) and this being a common ZCL specified frame, the *manufSpec* is set to zero.



This sub-field shall be set to TRUE for all manufacturer-specific extensions to ZCL specification (addition of new attributes, commands to existing clusters and addition of new clusters).

5.1.3 Commands

The HA reference application demonstrates usage of ZCL specified commands.

In #189, an addGroup command is sent from the Dimmer switch to the Light for the Group ID, defined in the application.

On receiving this command, the dimmable Light adds the Group ID and Group Name to its Group Table (maintained by the application).

It shall then generate an appropriate Add Group Response command indicating success or failure.

It is useful to group devices to send messages to the group as broadcast data.

In #333, an On command is sent from the Dimmer switch to the Light with the *noDefaultResp* sub-field in the ZCL header set to zero. This causes a default Response packet to be sent out from the light with the *statusCode* 0x00(SUCCESS).

Default Response is sent out under the following conditions:

- 1. The Disable default response bit of its Frame control field is set to 0(as discussed above).
- 2. No response command exists for the received command.
- 3. For an unsupported command received.

5.1.4 Security

By default, network layer encryption with the network key is used for cluster commands.

Table 5-2. Relevant Config. Server Parameters.

	Config. server parameter
Knowledge of security keys on device	CS_ZDO_SECURITY_STATUS
Active Network Key	CS_NETWORK_KEY

5.2 ZigBee Light Link

The ZLL Demo in the BitCloud Profile Suite for ZLL package implements standard device types: color scene controller, color light and bridge. The sniffer log referenced in this section is <code>ZLL_CL_CSR.wrk</code> which shall be viewed with Perytons Network Analyzer.

5.2.1 Touchlinking

The ZigBee Light Link network is not created by a coordinator. Instead, network parameters are transferred from an innetwork device to a factory-new device (device that is to join the network). This process of transferring network information occurs as Inter-PAN data exchange and is called touchlinking or commissioning.

In the sniffer log, packets #3 to #20 cover the touch-link procedure. All frames in this sequence follow the inter-PAN command format specified by [6].

Table 5-3.	Fields of Interest – Inter-PAN Command Frame Format.

Frame field	Value
MAC header -> Frame Control	panIDComp or Intra-PAN set to zero
NWK/APS header-> Frame Control	Frame Type set to reserved type
Inter-PAN Transaction Identifier	32-bit unique value valid throughout Touchlink process

The first part of the Touchlink process is the scanning done on the channel set (primary and if required, secondary) to search for devices in the vicinity.

To monitor activity on multiple channels at the same time, it is necessary that the sniffing tool is capable of multichannel capture and the number of hardware sniffing tools used shall equal the number of channels to be monitored. The log, *ZLL_CL_CSR.wrk*, was captured using a Perytons M-series Network Analyzer.

It is also possible to use BitCatcher and Wireshare interface and start capture on the first primary channel (0x0B) and identify the *logicalChannel* selected from the network start response frame payload.

5.2.2 Cluster Commands

The command and response frame transfer is as explained in Section 5.1.3.

5.2.3 Security

A ZLL network uses network layer security. Cluster data exchange can define security level in the cluster definition file.

A non-ZLL device when being commissioned into the ZLL network gets the network key securely using a pre-installed link key. The ZLL pre-installed link key is a secret shared by all certified ZLL devices and will be distributed only to certified manufacturers.

For development and testing purposes, the ZLL specification allows use of the keys in Table 5-4. These values can be found in file <code>N_Security_Calc.c</code> in directory <code>\zllplatform\zll\n_security\src</code> in the BitCloud for ZLL SDK package.

Table \$	5-4.	Security	Keys.
----------	------	----------	-------

Кеу	Value
Certification Network Key	ZLL_SECURITY_KEY "\xC0\xC1\xC2\xC3\xC4\xC5\xC6\xC7\xC8\xC9\xCA\xCB\xCC\xCD\xCE\xCF"
Certification pre-installed Link key	ZLL_PREINSTALLED_LINK_KEY "\xD0\xD1\xD2\xD3\xD4\xD5\xD6\xD7\xD8\xD9\xDA\xDB\xDC\xDD\xDE\xDF"

5.3 ZigBee Smart Energy

ZigBee Smart Energy networks employ similar data exchange mechanisms using clusters and commands as described in Section 5.1. Smart Energy Profile Security is different from the standard network security with link keys employed in Home Automation networks.

Smart Energy security requires the use of the Key Establishment cluster to generate a link key for secure data exchange between two devices in a SE network.

The sniffer log, *SE_Key_estb.dcf*, shows the packet exchange during the key establishment process which is called the Certificate Based Key Establishment (CBKE) process.

#17 shows that the device with short address 0x4810 has joined the network and authenticated to the Trust Center 0x0000 successfully using a pre-configured Trust Center Link key.

After service discovery for the KE cluster, #36 shows the transfer of the *Initiate Key Establishment* frame from 0x4810 to 0x000 and the response frame from the Trust Center. These frames contain the following items:

- Timing information of generation of the remaining KE packets. This is configurable as the cryptographic computation time may vary with devices
- Identity field containing the test Certificate of the device

Packets #55 and #64 show exchange of Ephemeral data which contains a randomly generated public key.

#73 and #83 show exchange of the Confirm Key request and response frames, these messages contain the derived /generated link key which will be used for further communication between the two devices.



6. Example Application Scenarios

The zip package that comes with this application note includes sniffer logs with the example application scenarios below. This will help the user to understand the packet exchange in the following cases that are not directly covered by the reference applications in the BitCloud SDK. These scenarios require the user to modify the application to see and understand the described scenarios.

6.1 PAN Same Channel Co-Existence

It is possible to have multiple ZigBee networks on the same channel. The sniffer log, *pan_same_channel_co-existence.dcf*, shows that it is possible for a second PAN to be started in the presence of an existing PAN.

Packet #0 to 426 shows data transfer in a Personal Area Network (PAN) with the short PANID 0x1AAB as can be seen from the macHeader->dstPanID field. This PAN consists of three devices, a coordinator, a router (macHeader->srcShortAddr =0x912F) and an end-device (macHeader->srcShortAddr =0x0B27).

The end-device transfers data to the router every two seconds.

#432 shows the MAC beacon request from an end-device to which the coordinator of 0x1AAA, coordinator and router of 0x1AAB respond.

Further, a router joins network 0x1AAB and the end-device transfers data to this router every two seconds.

6.2 End-to-End Establishment of Application Link Key

In a secure network, when two devices need to communicate on a secure link with each other, they shall request a link key from the Trust Center.

In Figure 6-1, Router 0xEA1E requests for a link key to Trust Center to communicate with router 0x12DC.



Figure 6-1. Link Key Establishment Between Two Routers
--

Packet view				
Clone 🔄 Edit				
€ zclHeader				
	port attributes (0x0A)			
Key : 47:D5:CF:9	E:3A:CD:19:67:3E:05:3C:FD:B1:40:7	2:80		
i⊒ nwkMic				
⊞∽mic				
Key : CC:CC:CC:C	C:CC:CC:CC:CC:CC:CC:CC:CC:CC:	cc;cc;cc		
⊕ parameters				
61 88 ED EC BF 1E EA D	DC 12 48 02 1E EA DC 12 0A CA	28 80 02 00 00 02 00 00 00 00 00 00 00 00 B3	CB 1C B4 E934 CD 7A C	IC 92222F99 72 88 1
61 88 ED EC BF 1E EA D Packets log	DC 12 48 02 1E EA DC 12 0A CA 3	28 80 02 00 00 02 00 00 00 00 00 00 00 00 B3	CB 1C B4 E934 CD 7A C	IC 92222F9972881
61 88 ED EC BF 1E EA D ackets log Cols All packets Time	DC 12 48 02 1E EA DC 12 0A CA	28 80 02 00 00 02 00 00 00 00 00 00 00 00 83 1	CB 1C B4 E9 34 CD 7A C	IC 92 22 2F 99 72 88 10
ackets log Cols All packets Time 7:59:56.765	DC 12 48 02 1E EA DC 12 0A CA 3	28 80 02 00 00 02 00 00 00 00 00 00 00 00 B3	CB 1C B4 E9 34 CD 7A C	IC 92 22 2F 99 72 88 10 MAC Dst
ackets log Cols All packets Time 7:59:56.765 7:59:56.765	DC 12 48 02 1E EA DC 12 0A CA 3	28 80 02 00 00 02 00 00 00 00 00 00 00 00 B3	CB 1C B4 E9 34 CD 7A C	IC 92 22 2F 99 72 88 1 MAC Dst
ackets log Cols All packets Time 7:59:56.765 7:59:56.774	DC 12 48 02 1E EA DC 12 0A CA Create Edit R Packet N 3666 3667 3668	28 80 02 00 00 02 00 00 00 00 00 00 00 00 B3 Remove Auto scroll Frame APS: Request Key Ack ZCL: Remove all scenes	CB 1C B4 E9 34 CD 7A C MAC Src 0xea1e 0xea1e	IC 92 22 2F 99 72 88 1 MAC Dst 0x0000 0x12dc
61 88 ED EC BF 1E EA D ackets log Cols All packets Time 17:59:56.765 17:59:56.774 17:59:56.774	DC 12 48 02 1E EA DC 12 0A CA Create Edit R Packet N 3666 3667 3668 3669	28 80 02 00 00 02 00 00 00 00 00 00 00 00 B3	CB 1C B4 E9 34 CD 7A C MAC Src Dxeale Dxeale	IC 92 22 2F 99 72 88 1 MAC Dst 0x0000 0x12dc
ackets log Cols All packets Time 7:59:56.765 7:59:56.774 7:59:56.774 7:59:56.774	DC 12 48 02 1E EA DC 12 0A CA 3	28 80 02 00 00 02 00 00 00 00 00 00 00 00 00	CB 1C B4 E9 34 CD 7A C MAC Src Dxeale Dxeale Dx0000	IC 92 22 2F 99 72 88 1 MAC Dst 0x0000 0x12dc 0xea1e
ackets log Cols All packets Time 7:59:56.765 7:59:56.774 7:59:56.774 7:59:56.774 7:59:56.782 7:59:56.782 7:59:56.782	DC 12 48 02 1E EA DC 12 0A CA 3	28 80 02 00 00 02 00 00 00 00 00 00 00 00 00	CB 1C B4 E9 34 CD 7A C MAC Src Dxeale Dxeale Dx0000	IC 92 22 2F 99 72 88 1 MAC Dst 0x0000 0x12dc 0xea1e
ackets log Cols All packets Time 7:59:56.765 7:59:56.774 7:59:56.774 7:59:56.782 7:59:56.782 7:59:56.789 7:59:56.789	C 12 48 02 1E EA DC 12 0A CA 3 Create Edit R Packet N 3666 3667 3668 3669 3670 3671 3672 3672 3672	28 80 02 00 00 02 00 00 00 00 00 00 00 00 00	CB 1C B4 E9 34 CD 7A C MAC Src 0xea1e 0xea1e 0x0000 0x12dc	IC 92 22 2F 99 72 88 1 MAC Dst Dx0000 0x12dc Dxea1e Dxea1e
61 88 ED EC BF 1E EA D ackets log Cols All packets Time 17:59:56.765 17:59:56.774 17:59:56.774 17:59:56.774 17:59:56.782 17:59:56.782 17:59:56.789 17:59:56.789 17:59:56.796	C 12 48 02 1E EA DC 12 0A CA 3 Create Edit F Packet N 3666 3667 3668 3669 3670 3671 3672 3673 3674	28 80 02 00 00 02 00 00 00 00 00 00 00 00 00 B3 Remove ✓ Auto scroll Frame APS: Request Key Ack 2CL: Remove all scenes Ack APS: Transport Key Ack APS: Transport Key Ack APS: Transport Key	CB 1C B4 E9 34 CD 7A C MAC Src 0xea1e 0xea1e 0x0000 0x12dc	IC 92 22 2F 99 72 88 1 MAC Dst Dx0000 Dx12dc Dxea1e Dxea1e Dxea1e
61 88 ED EC BF 1E EA D ackets log Cols All packets Time 17:59:56.765 17:59:56.774 17:59:56.774 17:59:56.782 17:59:56.782 17:59:56.789 17:59:56.796 17:59:56.796	C 12 48 02 1E EA DC 12 0A CA 3 Create Edit R 3666 3667 3668 3669 3670 3671 3672 3671 3672 3673 3674 3675	28 80 02 00 00 02 00 00 00 00 00 00 00 00 00 B3 Remove ✓ Auto scroll Frame APS: Request Key Ack 2CL: Remove all scenes Ack 2CL: Remove all scenes Ack APS: Transport Key Ack APS: Transport Key Ack APS: Transport Key	CB 1C B4 E9 34 CD 7A C MAC Src 0xea1e 0x0000 0x12dc 0x0000	IC 92 22 2F 99 72 88 1 MAC Dst Dx0000 Dx12dc Dxea1e Dxea1e Dxea1e Dx12dc
61 88 ED EC BF 1E EA D ackets log Cols All packets Time 77:59:56.765 77:59:56.774 77:59:56.774 77:59:56.782 77:59:56.782 77:59:56.789 77:59:56.789 77:59:56.796 77:59:56.796 77:59:56.817	C 12 48 02 1E EA DC 12 0A CA 3 Create Edit R Packet N 3666 3667 3668 3669 3670 3671 3672 3673 3674 3675 3676	28 80 02 00 00 02 00 00 00 00 00 00 00 00 00	CB 1C B4 E9 34 CD 7A C MAC Src Dxeale Dxeale Dx0000 Dx12dc Dx12dc	IC 92 22 2F 99 72 88 1 MAC Dst Dx0000 0x12dc Dxea1e 0x2dc Dxea1e
61 88 ED EC BF 1E EA D Packets log Cols All packets Time 77:59:56.765 77:59:56.774 77:59:56.774 77:59:56.782 77:59:56.782 77:59:56.789 77:59:56.789 77:59:56.796 77:59:56.796 77:59:56.817	C 12 48 02 1E EA DC 12 0A CA 3 Create Edit R Packet N 3666 3667 3668 3669 3670 3671 3672 3673 3674 3675 3675 3676 3677	28 80 02 00 00 02 00 00 00 00 00 00 00 00 00	CB 1C B4 E9 34 CD 7A C MAC Src 0xea1e 0x0000 0x12dc 0x0000 0x12dc	IC 92 22 2F 99 72 88 1 MAC Dst Dx0000 Dx12dc Dxea1e Dxea1e Dxea1e Dxea1e

Trust Center uses the Transport Key command to send the link key to both the routers. The link key is applied in further data exchange between the routers. #3678 uses this link, as seen in the packet view -> apsMic-> Key.

6.3 ZLL – ZHA Interoperability

It is possible for a ZigBee Home Automation device to join a ZLL network. The sniffer log, *ZLL_HA_S_L_GW.dcf*, shows an occupancy sensor device joining a ZLL network at packet #305.

It does a classical join and service discovery for the occupancy sensing cluster ID,0x0406. The Profile ID can be observed to be 0x0104 (Home Automation Profile).

The occupancy sensor then periodically reports the value of the occupancy sensing attribute to the ZLL light.

7. References

- [1]. Atmel AVR2050: BitCloud Developer Guide
- [2]. RZ600
- [3]. BitCatcher
- [4]. AVR2015: RZRAVEN Quick Start Guide
- [5]. BitCloud Profile Suite
- [6]. ZigBee Standards Documentation
- [7]. Atmel Wireshark Interface Atmel Gallery
- [8]. Wireshark
- [9]. Perytons Network Analyzer
- [10]. Dresden Elektronik Sniffer Tools



8. Revision History

Doc. Rev.	Date	Comments
32210A	09/2013	Initial document release



Atmel

Enabling Unlimited Possibilities®

Atmel Corporation

1600 Technology Drive San Jose, CA 95110 USA **Tel:** (+1)(408) 441-0311 **Fax:** (+1)(408) 487-2600 www.atmel.com

Atmel Asia Limited Unit 01-5 & 16, 19F BEA Tower, Millennium City 5

BEA Tower, Millennium City 418 Kwun Tong Road Kwun Tong, Kowloon HONG KONG Tel: (+852) 2245-6100 Fax: (+852) 2722-1369

Atmel Munich GmbH Business Campus Parkring 4 D-85748 Garching b. Munich GERMANY Tel: (+49) 89-31970-0 Fax: (+49) 89-3194621

Atmel Japan G.K.

16F Shin-Osaki Kangyo Building 1-6-4 Osaki, Shinagawa-ku Tokyo 141-0032 JAPAN **Tel:** (+81)(3) 6417-0300 **Fax:** (+81)(3) 6417-0370

© 2013 Atmel Corporation. All rights reserved. / Rev.: 32210A-WIRELESS-09/2013

Atmel[®], Atmel logo and combinations thereof, AVR[®], BitCloud[®], Enabling Unlimited Possibilities[®], and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Windows[®] is a registered trademark of Microsoft Corporation in U.S. and or other countries. Other terms and product names may be trademarks of others.

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.