ZigBee Light Link Profile Specification Version 1.0

ZigBee Document 11-0037-10	
April 5th, 2012	
Sponsored by: ZigBee Alliance	
Accepted by	This document has been accepted for release by the ZigBee Alliance Board of Directors
Abstract	This specification defines the protocol infrastructures and services available to applications operating on the ZigBee Pro platform using the Light Link profile.
Keywords	ZLL, consumer, residential, lighting, Light Link, profile.

This page is intentionally blank

Notice of use and disclosure

This page is intentionally blank

Revision history

Revision	Version	Date	Details	Editor
-	0.1	March 2010	First draft	Phil Jamieson
0	0.65	January 21st, 2011	Candidate ZigBee draft	Phil Jamieson
1	0.7	March 29th, 2011	Candidate 0.7 draft. Included sensor, security and minor fixes.	Phil Jamieson
2	0.7	June 2nd, 2011	Fixes from member comments, test events and the results of the network interoperability study.	Phil Jamieson
3	0.9	August 19th, 2011	Fixes from comments received during test events.	Phil Jamieson
4	0.9	October 5th, 2011	Resolved issues from and following the Paris test event. Added color temperature feature.	Phil Jamieson
5	0.9	November 18th, 2011	Resolved issues from and following the October, 2011 Hull test event.	Phil Jamieson
6	0.9	December 13th, 2011	Resolved issues from the November/December, 2011 Hull test event.	Phil Jamieson
7	1.0	February 13th, 2012	Resolved issues from the v0.9 letter ballot and others raised on the reflector.	Phil Jamieson
8	1.0	February 27th, 2012	Resolved issues from the February, 2012 Hull test event.	Phil Jamieson
9	1.0	March 26th, 2012	Resolved minor issues raised on the reflector.	Phil Jamieson
10	1.0	April 5th, 2012	Updated boilerplate following board approval.	Phil Jamieson

This page is intentionally blank

Table of Contents

This page is intentionally blank

List of Figures

List of Tables

1 Introduction

This document specifies the ZigBee Light Link (ZLL) profile and describes the expected behavior of the devices that need to operate in a ZLL network.

The ZigBee Light Link profile addresses devices and functionality in the over-the-counter, consumer lighting application domain. It is based on ZigBee-Pro and utilizes the clusters defined in the ZigBee Cluster Library. In addition, it provides a commissioning mechanism which will give the consumer a simple and intuitive experience when connecting devices together.

The commissioning mechanism is known as touch-linking. Touch-linking is a method of finding devices in the neighborhood based on received signal strength. A touch-link action is easy for the user to understand and can replace buttons on a device that would otherwise be required to facilitate commissioning.

The touch-link operation is divided into two parts; device discovery and transferring network settings. The result of device discovery is a list of device information which includes network capabilities, device type and whether a device is factory-new. It is up to the application to take further action on one or more devices found.

ZigBee relies heavily on a network device called the coordinator. Such a coordinator is assumed to be always available and therefore always powered. In a lighting system intended for the home in a consumer market, this is a heavy constraint. For this reason, another method was found to benefit from the ZigBee network functionality (especially routing) without the need of having a coordinator. This leads to other ways of starting a network, joining devices, assigning network addresses and applying security.

The additional procedures for touch-linking and other network operations heavily rely on inter-PAN messages. Inter-PAN messages can be transferred between devices that are on the same channel. A special flag in the message indicates to by-pass the ZigBee Network layer and for this reason, the two devices do not need to be part of the same ZigBee network.

1.1 Scope

The scope of the ZLL profile is as follows:

- It is intended for ZigBee applications in consumer lighting.
- There will be some (but limited) overlapping applications with residential installed home automation and smart energy profiles.
- It is intended to be built on the ZigBee-Pro stack.
- It is intended to be natively interoperable with other ZigBee profiles.
- It is not initially intended for advanced residential, commercial building, industrial or professional outdoor lighting networks.

1.2 Purpose

The purpose of the ZLL profile is as follows:

- To provide an evolutionary consumer experience for lighting devices in which further purchases enhance the overall system.
- To develop a simple and sensible ZigBee specification for over-the-counter lamps and luminaries in the consumer market space.
- To develop a solution, fully in line with consumer market boundary conditions on commissioning, security, ease of use, network scale, cost, etc.
- To be able to address non-installer consumer lighting related features which are not addressed in the ZHA profile.

1.3 Conformance levels

The following words, used throughout this document, have specific meanings:

May A key word indicating a course of action permissible within the limits of the standard (*may* equals *is permitted*).

Shall A key word indicating mandatory requirements to be strictly followed in order to conform to the standard; deviations from shall are prohibited (*shall* equals *is required to*).

Should A key word indicating that, among several possibilities, one is

recommended as particularly suitable, without mentioning or excluding others; that a certain course of action is preferred but not necessarily required; or, that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

1.4 Conventions

1.4.1 Number formats

In this specification hexadecimal numbers are prefixed with the designation "0x" and binary numbers are prefixed with the designation "0b". All other numbers are assumed to be decimal unless indicated otherwise within the associated text.

1.4.2 Transmission order

The frames in this specification are described as a sequence of fields in a specific order. All frame formats are depicted in the order in which they are transmitted by the PHY, from left to right (e.g. Figure 15) where the leftmost bit is transmitted first in time or top to bottom (e.g. Figure 44) where the topmost bit is transmitted first in time. Bits within each field are numbered from 0 (leftmost and least significant) to k-1 (rightmost and most significant), where the length of the field is k bits. Fields that are longer than a single octet are sent to the MAC in the order from the octet containing the lowest numbered bits to the octet containing the highest numbered bits.

1.4.3 Reserved values

Unless otherwise specified, all reserved fields appearing in a frame structure (c.f. 7.1.2.3.1.4) shall be set to zero on transmission and ignored upon reception. Reserved values appearing in multi-value fields (c.f. 7.1.2.3) shall not be used.

2 References[1]

2.1 ZigBee Alliance documents

[R1] ZigBee Specification, ZigBee Alliance document 053474.
[R2] ZigBee Cluster Library Specification, ZigBee Alliance document 075123.
[R3] ZigBee Smart Energy Profile Specification, ZigBee Alliance document 075356.
[R4] ZigBee-2007 Layer PICS and Stack Profiles, ZigBee Alliance document 08006.
[R5] ZigBee Light Link Security Agreement, ZigBee Alliance document 11-5398.
[R6] ZigBee Light Link Test Specification, ZigBee Alliance document 11-0039.

2.2 Other documents

[R7] Recommendation for Block Cipher Modes of Operation. Methods and Techniques, NIST Special Publication 800-38A.

[1] The version and date information in these references was correct at the time this document was released.

3 Definitions

Coordinator	The ZigBee node responsible for starting a network and allowing other devices to join this network in a secure way. A coordinator is also a router.
Device	Product implementation of the ZLL profile.
End-device	A ZigBee node which has no capability of routing messages through the network.
Endpoint	A ZigBee endpoint implements application features that are non-networking related (which the exception of the mandatory endpoint 0 which handles the node’s network management functions).
Factory New	The device does not contain any network parameters and is not part of a network. When a device is reset to factory new, its network parameters are erased.
IEEE Address	An 8-byte unique address. Sometimes also referred to as the MAC address.
Master Key	128 bit security key used to protect the transmission of the Network Key. The ZLL Master Key is distributed to ZigBee members that want to commercialize one or more ZLL devices. Pre-requisites are a successful (pre-) certification of a ZLL device and signing the ZLL Master Key License Agreement by the ZigBee member. The ZLL Master Key is identical for each ZLL profile implementer.
Master Key License Agreement	License agreement that describes the ZLL safekeeping rules for the ZLL Master Key
Network Parameters	Set of extended PAN ID, PAN ID, channel number, network update ID, network address and network key.
Node	A collection of independent device descriptions and applications residing in a single unit and sharing a common IEEE 802.15.4 radio.
Router	A ZigBee node capable of routing messages through the network and acting as a parent for end-devices.
Sub-device	A device may be divided in sub-devices when it has more application endpoints, for example two independent light outputs.
Touch-link	The user operation of holding one device (e.g., a remote controller) physically close to another device (e.g., a light) in order to facilitate a network connection.

4 Acronyms and abbreviations

AES	Advanced encryption standard
ECB	Electronic code book mode of block-cipher operation
ED	End device
EP	Endpoint
FN	Factory new
ID	Identifier
IEEE	Institute of electrical and electronic engineers
LQI	Link quality indication
MAC	Medium access control
NFN	Non factory new
NIB	Network information base
NLDE	Network layer data entity
NLME	Network layer management entity
NVRAM	Non-volatile random access memory
NWK	Network
PAN	Personal area network
PHY	Physical
PIU	Plug-in unit
POS	Personal operating space
R	Router
RF	Radio frequency
RSSI	Received signal strength indicator
SAP	Service access point
ZCL	ZigBee cluster library
ZDO	ZigBee device objects
ZHA	ZigBee home automation
ZLL	ZigBee Light Link

5 Device descriptions

All ZLL devices shall include the ZLL commissioning cluster (see 7.1) as a server, a client or both server and client. Table 1 lists the device descriptions for the devices defined in the ZLL profile along with their respective device identifiers and ZLL commissioning cluster orientations. These devices are described in the following sub-clauses.

Note that devices marked (♣) do not need to support the commissioning utility functionality of the ZLL commissioning cluster.

The device IDs are used to identify the application on a certain device and is the primary means to know which clusters are used by that application

Table 1 – Device descriptions for the application device component

	Device description	**Device ID**	**Commissioning server**	**Commissioning server/client**	**Commissioning client**
Lighting devices	On/off light (♣)	0x0000	✓	✓	×
	On/off plug-in unit (♣)	0x0010	✓	✓	×
	Dimmable light (♣)	0x0100	✓	✓	×
	Dimmable plug-in unit (♣)	0x0110	✓	✓	×
	Color light (♣)	0x0200	✓	✓	×
	Extended color light (♣)	0x0210	✓	✓	×
	Color temperature light (♣)	0x0220	✓	✓	×
Controller devices	Color controller	0x0800	✓	✓	✓
	Color scene controller	0x0810	✓	✓	✓
	Non-color controller	0x0820	✓	✓	✓
	Non-color scene controller	0x0830	✓	✓	✓
	Control bridge	0x0840	✓	✓	✓
	On/off sensor	0x0850	✓	✓	✓

All other values in the range 0x0000-0xffff are reserved for future use.

5.1 Common clusters

Support for certain clusters is common for all devices in this profile. All devices shall support the clusters listed in Figure 1.

All devices	
Server clusters	**Client clusters**
Basic	-

Figure 1 – Clusters common to all devices

5.2 Lighting devices

5.2.1 On/off light

In addition to those clusters supported in Figure 1, the on/off light device shall support the clusters listed in Figure 2.

On/off light [Device ID: 0x0000]	
Server clusters	**Client clusters**
Identify	-
Groups	
Scenes	
On/off	

Figure 2 – Additional clusters supported by the on/off light

5.2.2 On/off plug-in unit

In addition to those clusters supported in Figure 1, the on/off plug-in unit device shall support the clusters listed in Figure 3.

On/off plug-in unit [Device ID: 0x0010]	
Server clusters	**Client clusters**
Identify	-
Groups	
Scenes	
On/off	

Figure 3 – Additional clusters supported by the on/off plug-in unit

5.2.3 Dimmable light

In addition to those clusters supported in Figure 1, the dimmable light device shall support the clusters listed in Figure 4.

Dimmable light [Device ID: 0x0100]	
Server clusters	**Client clusters**
Identify	-
Groups	
Scenes	
On/off	
Level control	

Figure 4 – Additional clusters supported by the dimmable light

5.2.4 Dimmable plug-in unit

In addition to those clusters supported in Figure 1, the dimmable plug-in unit device shall support the clusters listed in Figure 5.

Dimmable plug-in unit [Device ID: 0x0110]	
Server clusters	**Client clusters**
Identify	-
Groups	
Scenes	
On/off	
Level control	

Figure 5 – Additional clusters supported by the dimmable plug-in unit

5.2.5 Color light

In addition to those clusters supported in Figure 1, the color light device shall support the clusters listed in Figure 6.

Color light [Device ID: 0x0200]	
Server clusters	**Client clusters**
Identify	-
Groups	
Scenes	
On/off	
Level control	
Color control	

Figure 6 – Additional clusters supported by the color light

For this device, in the *color control* cluster, the *ColorCapabilities* attribute shall be set to 0x000f, indicating support for hue/saturation, enhanced hue, color loop and XY.

5.2.6 Extended color light

In addition to those clusters supported in Figure 1, the extended color light device shall support the clusters listed in Figure 7.

Extended color light [Device ID: 0x0210]	
Server clusters	**Client clusters**
Identify	-
Groups	
Scenes	
On/off	
Level control	
Color control	

Figure 7 – Additional clusters supported by the extended color light

For this device, in the *color control* cluster, the *ColorCapabilities* attribute shall be set to 0x001f, indicating support for hue/saturation, enhanced hue, color loop, XY and color temperature.

5.2.7 Color temperature light

In addition to those clusters supported in Figure 1, the color temperature light device shall support the clusters listed in Figure 8.

Color temperature light [Device ID: 0x0220]	
Server clusters	**Client clusters**
Identify	-
Groups	
Scenes	
On/off	
Level control	
Color control	

Figure 8 – Additional clusters supported by the color temperature light

For this device, in the *color control* cluster, the *ColorCapabilities* attribute shall be set to 0x0010, indicating support for color temperature.

5.3 Controller devices

5.3.1 Color controller

In addition to those clusters supported in Figure 1, the color controller device shall support the clusters listed in Figure 9.

Color controller [Device ID: 0x0800]	
Server clusters	**Client clusters**
	Identify
	Groups
	On/off
	Level control
	Color control
ZLL commissioning: utility	ZLL commissioning: utility

Figure 9 – Additional clusters supported by the color controller

5.3.2 Color scene controller

In addition to those clusters supported in Figure 1, the color scene controller device shall support the clusters listed in Figure 10.

Color scene controller [Device ID: 0x0810]	
Server clusters	**Client clusters**
	Identify
	Groups
	Scenes
	On/off
	Level control
	Color control
ZLL commissioning: utility	ZLL commissioning: utility

Figure 10 – Additional clusters supported by the color scene controller

5.3.3 Non-color controller

In addition to those clusters supported in Figure 1, the non-color controller device shall support the clusters listed in Figure 11.

Non-color controller [Device ID: 0x0820]	
Server clusters	**Client clusters**
	Identify
	Groups
	On/off
	Level control
ZLL commissioning: utility	ZLL commissioning: utility

Figure 11 – Additional clusters supported by the non-color controller

5.3.4 Non-color scene controller

In addition to those clusters supported in Figure 1, the non-color scene controller device shall support the clusters listed in Figure 12.

Non-color scene controller [Device ID: 0x0830]	
Server clusters	**Client clusters**
	Identify
	Groups
	Scenes
	On/off
	Level control
ZLL commissioning: utility	ZLL commissioning: utility

Figure 12 – Additional clusters supported by the non-color scene controller

5.3.5 Control bridge

In addition to those clusters supported in Figure 1, the control bridge device shall support the clusters listed in Figure 13.

Control bridge [Device ID: 0x0840]	
Server clusters	**Client clusters**
	Identify
	Groups
	Scenes
	On/off
	Level control
	Color control
ZLL commissioning: utility	ZLL commissioning: utility

Figure 13 – Additional clusters supported by the control bridge

5.3.6 On/off sensor

In addition to those clusters supported in Figure 1, the on/off sensor device shall support the clusters listed in Figure 14.

On/off sensor [Device ID: 0x0850]	
Server clusters	**Client clusters**
	Identify
	Groups
	Scenes
	On/off
	Level control
	Color control
ZLL commissioning: utility	ZLL commissioning: utility

Figure 14 – Additional clusters supported by the on/off sensor

6 ZCL usage and enhancements

In the ZCL, each cluster has a number of attributes and commands which may be mandatory or optional. In this profile, however, optional attributes or commands are discouraged.

If two devices support a different set of clusters, attributes or commands at their ZCL server side, they shall have different device identifiers.

Attribute reporting shall not be used in this profile.

6.1 General command frames

A ZLL device shall implement each of the general command frames listed in Table 2.

Table 2 – Usage of general command frames

Command identifier field value	Description	Reference	Usage	
			Tx	Rx
0x00	Read attributes	[R2] 2.4.1	Optional	Mandatory
0x01	Read attributes response	[R2] 2.4.2	Mandatory	Mandatory is read attributes transmission is supported.
0x02	Write attributes	[R2] 2.4.3	Optional	Mandatory
0x03	Write attributes undivided	[R2] 2.4.4	Optional	Mandatory
0x04	Write attributes response	[R2] 2.4.5	Mandatory	Mandatory is write attributes or write attributes undivided transmission is supported.
0x05	Write attributes no response	[R2] 2.4.6	Optional	Mandatory
0x0b	Default response	[R2] 2.4.12	Mandatory	Mandatory

6.2 Basic cluster

6.2.1 Server

6.2.1.1 Attributes

When a device implements the *basic* cluster at the ZCL server side, it shall support the attributes listed in Table 3.

Table 3 – Mandatory attributes of the basic cluster

Identifier	Name
0x0000	ZCLVersion
0x0001	ApplicationVersion
0x0002	StackVersion
0x0003	HWVersion
0x0004	ManufacturerName
0x0005	ModelIdentifier
0x0006	DateCode
0x0007	PowerSource

In addition, the server side of the *basic* cluster shall be enhanced with the attributes listed in Table 4.

Table 4 – Additional attributes of the server side of the basic cluster

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x4000	SWBuildID	Character string	Up to 16 bytes	Read only	Empty string	Mandatory

6.2.1.1.1 SWBuildID attribute

The *SWBuildID* attribute represents a detailed, manufacturer-specific reference to the version of the software.

6.2.1.2 Commands received

No cluster specific commands are received by the server.

6.2.1.3 Commands generated

No cluster specific commands are generated by the server.

6.2.2 Client

6.2.2.1 Attributes

The client has no attributes.

6.2.2.2 Commands received

No cluster specific commands are received by the client.

6.2.2.3 Commands generated

No cluster specific commands are generated by the client.

6.3 Identify cluster

6.3.1 Server

6.3.1.1 Attributes

When a device implements the *identify* cluster at the ZCL server side, it shall support the attributes listed in Table 5.

Table 5 – Mandatory attributes of the identify cluster

Identifier	**Name**
0x0000	IdentifyTime

6.3.1.2 Commands received

When a device implements the *identify* cluster at the ZCL server side, it shall be able to receive the commands listed in Table 6.

Table 6 – Mandatory commands received by the server side of the identify cluster

Identifier	**Name**
0x00	Identify
0x01	Identify query

In addition, the server side of the *identify* cluster shall be enhanced to be able to receive the commands listed in Table 7.

Table 7 – Additional commands received by the server side of the identify cluster

Command identifier field value	**Description**	**Mandatory/ Optional**
0x40	Trigger effect	Mandatory

6.3.1.2.1 Trigger effect command

The *trigger effect* command allows the support of feedback to the user, such as a certain light effect.

The payload of this command shall be formatted as illustrated in Figure 15.

Octets	1	1
Data type	Unsigned 8-bit integer	Unsigned 8-bit integer
Field name	Effect identifier	Effect variant

Figure 15 – Format of the trigger effect command

6.3.1.2.1.1 Effect identifier field

The *effect identifier* field is 8-bits in length and specifies the identify effect to use. This field shall contain one of the non-reserved values listed in Table 8.

Table 8 – Values of the effect identifier field of the trigger effect command

***effect identifier* field value**	**Effect**[2]	**Notes**
0x00	Blink	E.g., Light is turned on/off once.
0x01	Breathe	E.g., Light turned on/off over 1 second and repeated 15 times.
0x02	Okay	E.g., Colored light turns green for 1 second; non-colored light flashes twice.
0x0b	Channel change	E.g., Colored light turns orange for 8 seconds; non-colored light switches to maximum brightness for 0.5s and then minimum brightness for 7.5s.
0xfe	Finish effect	Complete the current effect sequence before terminating. E.g., if in the middle of a breathe effect (as above), first complete the current 1s breathe effect and then terminate the effect.
0xff	Stop effect	Terminate the effect as soon as possible.
All other values	Reserved	-

6.3.1.2.1.2 Effect variant field

The *effect variant* field is 8-bits in length and is used to indicate which variant of the effect, indicated in the *effect identifier* field, should be triggered. If a device does not support the given variant, it shall use the default variant. This field shall contain one of the non-reserved values listed in Table 9.

Table 9 – Values of the effect variant field of the trigger effect command

effect variant field value	**Description**
0x00	Default
0x01 – 0xff	Reserved

6.3.1.2.1.3 Effect on receipt

On receipt of this command, the device shall execute the trigger effect indicated in the *effect identifier* and *effect variant* fields. If the *effect variant* field specifies a variant that is not supported on the device, it shall execute the default variant.

6.3.1.3 Commands generated

When a device implements the *identify* cluster at the ZCL server side, it shall be able to generate the commands listed in Table 10.

[2] Implementers should indicate during testing how they handle each effect.

Table 10 – Mandatory commands generated by the server side of the identify cluster

Identifier	Name	Permitted transmission services		
		Unicast	Groupcast	Broadcast
0x00	Identify query response	✓	×	×

6.3.2 Client

6.3.2.1 Attributes

The client has no attributes.

6.3.2.2 Commands received

The client receives the cluster specific response commands listed in Table 11. These commands are detailed in 6.3.1.3.

Table 11 – Commands received by the client side of the identify cluster

Identifier	Name	Mandatory on transmission of
0x00	Identify query response	Identify query

6.3.2.3 Commands generated

The client generates the cluster specific commands listed in Table 12, as required by the application. These commands are detailed in 6.3.1.2.

Table 12 – Commands generated by the client side of the identify cluster

Identifier	Name	Usage	Permitted transmission services		
			Unicast	Groupcast	Broadcast
0x00	Identify	Optional	✓	✓	✓
0x01	Identify query	Optional	✓	✓	✓
0x40	Trigger effect	Optional	✓	✓	✓

6.4 Groups cluster

6.4.1 Server

6.4.1.1 Attributes

When a device implements the *groups* cluster at the ZCL server side, it shall support the attributes listed in Table 13.

Table 13 – Mandatory attributes of the groups cluster

Identifier	Name	Comments
0x0000	NameSupport	Fixed to 0, indicating no name support.

6.4.1.2 Commands received

When a device implements the *groups* cluster at the ZCL server side, it shall be able to receive the commands listed in Table 14.

Table 14 – Mandatory commands received by the server side of the groups cluster

Identifier	Name
0x00	Add group
0x01	View group
0x02	Get group membership
0x03	Remove group
0x04	Remove all groups
0x05	Add group if identifying

6.4.1.2.1 Generic usage notes

On receipt of the *add group*, *view group*, *get group membership* or *remove group* command frames via the groupcast or broadcast transmission service, no response shall be given.

6.4.1.3 Commands generated

When a device implements the *groups* cluster at the ZCL server side, it shall be able to generate the commands listed in Table 15.

Table 15 – Mandatory commands generated by the server side of the groups cluster

Identifier	Name	Permitted transmission services		
		Unicast	Groupcast	Broadcast
0x00	Add group response	✓	✗	✗
0x01	View group response	✓	✗	✗
0x02	Get group membership response	✓	✗	✗
0x03	Remove group response	✓	✗	✗

6.4.2 Client

6.4.2.1 Attributes

The client has no attributes.

6.4.2.2 Commands received

The client receives the cluster specific response commands listed in Table 16. These commands are detailed in 6.4.1.3.

Table 16 – Commands received by the client side of the groups cluster

Identifier	Name	Mandatory on transmission of
0x00	Add group response	Add group
0x01	View group response	View group
0x02	Get group membership response	Get group membership
0x03	Remove group response	Remove group

6.4.2.3 Commands generated

The client generates the cluster specific commands listed in Table 17, as required by the application. These commands are detailed in 6.4.1.2.

Table 17 – Commands generated by the client side of the groups cluster

Identifier	Name	Usage	Permitted transmission services		
			Unicast	Groupcast	Broadcast
0x00	Add group	Optional	✓	✓	✓
0x01	View group	Optional	✓	✓	✓
0x02	Get group membership	Optional	✓	✓	✓
0x03	Remove group	Optional	✓	✓	✓
0x04	Remove all groups	Optional	✓	✓	✓
0x05	Add group if identifying	Optional	✓	✓	✓

6.5 Scenes cluster

6.5.1 Server

6.5.1.1 Attributes

When a device implements the *scenes* cluster at the ZCL server side, it shall support the attributes listed in Table 18.

Table 18 – Mandatory attributes of the scenes cluster

Identifier	Name	Comments
0x0000	SceneCount	-
0x0001	CurrentScene	-
0x0002	CurrentGroup	-
0x0003	SceneValid	-
0x0004	NameSupport	Fixed to 0, indicating no name support.

6.5.1.2 Scene table enhancements

When a device implements the *scenes* cluster at the ZCL server side, it shall ensure the attributes listed in Table 19 are added to the scene table.

Table 19 – Scenes cluster enhancements to the scene table

Field	Type	Valid range	Description
TransitionTime100ms	Unsigned 8-bit integer	0x00 – 0x09	Together with the scene transition time element, this allows the transition time to be specified in tenths of a second.

6.5.1.3 Commands received

When a device implements the *scenes* cluster at the ZCL server side, it shall be able to receive the commands listed in Table 20.

Table 20 – Mandatory commands received by the server side of the scenes cluster

Identifier	Name
0x00	Add scene
0x01	View scene
0x02	Remove scene
0x03	Remove all scenes
0x04	Store scene
0x05	Recall scene
0x06	Get scene membership

In addition, the server side of the *scenes* cluster shall be enhanced to be able to receive the commands listed in Table 21.

Table 21 – Additional commands received by the server side of the scenes cluster

Command identifier field value	Description	Mandatory/ Optional
0x40	Enhanced add scene	Mandatory
0x41	Enhanced view scene	Mandatory
0x42	Copy scene	Mandatory

6.5.1.3.1 Generic usage notes

Scene identifier 0x00, along with group identifier 0x0000, is reserved for the global scene used by the *OnOff* cluster.

On receipt of the *add scene*, *view scene*, *remove scene*, *remove all scenes*, *store scene*, *get scene membership*, *enhanced add scene*, *enhanced view scene* or *copy scene* command frames via the groupcast or broadcast transmission service, no response shall be given.

On receipt of the *add scene* command, the *scene transition time* element of the scene table shall be updated with the value of the *transition time* field and the *TransitionTime100ms* element shall be set to zero.

6.5.1.3.2 Enhanced add scene command

The *enhanced add scene* command allows a scene to be added using a finer scene transition time that was available in the ZCL.

The payload of this command shall be formatted in the same way as the *add scene* command, specified in the ZCL *scenes* cluster, with the following enhancements:

1. The *transition time* field shall be measured in tenths of a second rather than in seconds.
2. The *extension fields* shall include the extensions defined in this specification.

6.5.1.3.2.1 Effect on receipt

On receipt of this command, the device shall (if possible) create an entry in the scene table with fields copied from the command payload. If there is already a scene in the table with the same *scene identifier* and *group identifier*, it shall overwrite it, (i.e., it shall first remove all information included in the original scene entry).

The *transition time* (measured in tenths of a second) shall be separated into whole seconds for the standard ZCL *transition time* field of the scene table entry and the new *TransitionTime100ms* field, as specified in this specification.

If the *enhanced add scene* command was received via the unicast data service, the *scenes* cluster server shall then generate and transmit an *enhanced add scene response* command back to the originator. Otherwise, no response shall be given.

6.5.1.3.3 Enhanced view scene command

The *enhanced view scene* command allows a scene to be retrieved using a finer scene transition time that was available in the ZCL.

The payload of this command shall be formatted in the same way as the *view scene* command, specified in the ZCL scenes cluster.

6.5.1.3.3.1 Effect on receipt

On receipt of this command, the device shall generate an appropriate *enhanced view scene response* command (see 6.5.1.4.2).

6.5.1.3.4 Copy scene command

The *copy scene* command allows a device to efficiently copy scenes from one group/scene identifier pair to another group/scene identifier pair.

The payload of this command shall be formatted as illustrated in Figure 16.

Octets	1	2	1	2	1
Data type	Unsigned 8-bit integer	Unsigned 16-bit integer	Unsigned 8-bit integer	Unsigned 16-bit integer	Unsigned 8-bit integer
Field name	Mode	Group identifier from	Scene identifier from	Group identifier to	Scene identifier to

Figure 16 – Format of the copy scene command

6.5.1.3.4.1 Mode field

The *mode* field is 8-bits in length and contains information of how the scene copy is to proceed. This field shall be formatted as illustrated in Figure 17.

Bits: 0	**Bits: 1-7**
Copy all scenes	Reserved

Figure 17 – Format of the mode field of the copy scene command

The *copy all scenes* subfield is 1-bit in length and indicates whether all scenes are to be copied. If this value is set to 1, all scenes are to be copied and the *scene identifier from* and *scene identifier to* fields shall be ignored. Otherwise this field is set to 0.

6.5.1.3.4.2 Group identifier from field

The *group identifier from* field is 16-bits in length and specifies the identifier of the group from which the scene is to be copied. Together with the *scene identifier from* field, this field uniquely identifies the scene to copy from the scene table.

6.5.1.3.4.3 Scene identifier from field

The *scene identifier from* field is 8-bits in length and specifies the identifier of the scene from which the scene is to be copied. Together with the *group identifier from* field, this field uniquely identifies the scene to copy from the scene table.

6.5.1.3.4.4 Group identifier to field

The *group identifier to* field is 16-bits in length and specifies the identifier of the group to which the scene is to be copied. Together with the *scene identifier to* field, this field uniquely identifies the scene to copy to the scene table.

6.5.1.3.4.5 Scene identifier to field

The *scene identifier to* field is 8-bits in length and specifies the identifier of the scene to which the scene is to be copied. Together with the *group identifier to* field, this field uniquely identifies the scene to copy to the scene table.

6.5.1.3.4.6 Effect on receipt

On receipt of the *copy scene* command, if the *copy all scenes* sub-field of the *mode* field is set to 1, the *scenes* cluster server shall copy all its available scenes with group identifier equal to the *group identifier from* field under the group identifier specified in the *group identifier to* field, leaving the scene identifiers the same. In this case, the *scene identifier from* and *scene identifier to* fields are ignored.

If a scene already exists under the same group/scene identifier pair, it shall be overwritten.

If the *copy scene* command was received via the unicast data service, the *scenes* cluster server shall then generate and transmit a *copy scene response* command back to the originator. Otherwise, no response shall be given.

6.5.1.4 Commands generated

When a device implements the *scenes* cluster at the ZCL server side, it shall be able to generate the commands listed in Table 22.

Table 22 – Mandatory commands generated by the server side of the scenes cluster

Identifier	Name	Permitted transmission services		
		Unicast	Groupcast	Broadcast
0x00	Add scene response	✓	✗	✗
0x01	View scene response	✓	✗	✗
0x02	Remove scene response	✓	✗	✗
0x03	Remove all scenes response	✓	✗	✗
0x04	Store scene response	✓	✗	✗
0x06	Get scene membership response	✓	✗	✗

In addition, the server side of the *scenes* cluster shall be enhanced to be able to generate the commands listed in Table 23.

Table 23 – Additional commands generated by the server side of the scenes cluster

Command identifier field value	Description	Mandatory/ Optional	Permitted transmission services		
			Unicast	Groupcast	Broadcast
0x40	Enhanced add scene response	Mandatory	✓	✗	✗
0x41	Enhanced view scene response	Mandatory	✓	✗	✗
0x42	Copy scene response	Mandatory	✓	✗	✗

6.5.1.4.1 Enhanced add scene response command

The *enhanced add scene response* command allows a device to respond to an *enhanced add scene* command.

The payload of this command shall be formatted in the same way as the *add scene response* command, specified in the ZCL scenes cluster.

6.5.1.4.2 Enhanced view scene response command

The *enhanced view scene response* command allows a device to respond to an *enhanced view scene* command using a finer scene transition time that was available in the ZCL.

The payload of this command shall be formatted in the same way as the *view scene response* command, specified in the ZCL scenes cluster, with the following enhancements:

1. The *transition time* field shall be measured in tenths of a second rather than in seconds.
2. The *extension fields* shall include the extensions defined in this specification.

6.5.1.4.2.1 When generated

The *enhanced view scene response* command is generated in response to a received *enhanced view scene* command. The entry in the scene table with scene identifier and group identifier given in the received *enhanced view scene* command is located (if possible). The *status* field is set to SUCCESS, NOT_FOUND (the scene is not present in the scene table) or INVALID_FIELD (the group is not present in the group table) as appropriate. The group identifier and scene identifier fields are set to the corresponding fields in the received *enhanced view scene* command.

If the status is SUCCESS, the *transition time*, *scene name* and *extension field* fields are copied from the corresponding fields in the table entry, otherwise they are omitted.

The *transition time* (measured in tenths of a second) shall be calculated from the standard transition time field of the scene table entry (measured in seconds) and the new *TransitionTime100ms* field, as specified in this specification.

6.5.1.4.3 Copy scene response command

The *copy scene response* command allows a device to respond to a *copy scene* command.

The payload of this command shall be formatted as illustrated in Figure 18.

Octets	1	2	1
Data type	Unsigned 8-bit integer	Unsigned 16-bit integer	Unsigned 8-bit integer
Field name	Status	Group identifier from	Scene identifier from

Figure 18 – Format of the copy scene response command

6.5.1.4.3.1 Status field

The *status* field is 8-bits in length and shall contain the status of the copy scene attempt. This field shall be set to one of the non-reserved values listed in Table 24.

Table 24 – Values of the status field of the copy scene response command

Status field value[3]	Description
SUCCESS	Success
INVALID_FIELD	Invalid scene specified
INSUFFICIENT_SPACE	Insufficient space in the scene table

[3] See [R2] for an enumerated list of status values.

6.5.1.4.3.2 Group identifier from field

The *group identifier from* field is 16-bits in length and specifies the identifier of the group from which the scene was copied, as specified in the *copy scene* command. Together with the *scene identifier from* field, this field uniquely identifies the scene that was copied from the scene table.

6.5.1.4.3.3 Scene identifier from field

The *scene identifier from* field is 8-bits in length and specifies the identifier of the scene from which the scene was copied, as specified in the *copy scene* command. Together with the *group identifier from* field, this field uniquely identifies the scene that was copied from the scene table.

6.5.1.4.3.4 When generated

The *copy scene response* command is generated in response to a received *copy scene* command. If, during the copy, there is no more space in the scene table for the entire next scene to be copied, the *status* field shall be set to INSUFFICIENT_SPACE and the scenes already copied shall be kept. If the group identifier from and scene identifier from fields do not specify a scene that exists in the scene table, the Status field shall be set to INVALID_FIELD. Otherwise, if the copy was successful, the *status* field shall be set to SUCCESS. The group identifier from and scene identifier from fields shall be set to the same values as in the corresponding fields of the received *copy scene* command.

6.5.2 Client

6.5.2.1 Attributes

The client has no attributes.

6.5.2.2 Commands received

The client receives the cluster specific response commands listed in Table 25. These commands are detailed in 6.5.1.4.

Table 25 – Commands received by the client side of the scenes cluster

Identifier	Name	Mandatory on transmission of
0x00	Add scene response	Add scene
0x01	View scene response	View scene
0x02	Remove scene response	Remove scene
0x03	Remove all scenes response	Remove all scenes
0x04	Store scene response	Store scene
0x06	Get scene membership response	Get scene membership
0x40	Enhanced add scene response	Enhanced add scene
0x41	Enhanced view scene response	Enhanced view scene
0x42	Copy scene response	Copy scene

6.5.2.3 Commands generated

The client generates the cluster specific commands listed in Table 26, as required by the application. These commands are detailed in 6.5.1.3.

Table 26 – Commands generated by the client side of the scenes cluster

Identifier	Name	Usage	Permitted transmission services		
			Unicast	Groupcast	Broadcast
0x00	Add scene	Optional	✓	✓	✓
0x01	View scene	Optional	✓	✓	✓
0x02	Remove scene	Optional	✓	✓	✓
0x03	Remove all scenes	Optional	✓	✓	✓
0x04	Store scene	Optional	✓	✓	✓
0x05	Recall scene	Optional	✓	✓	✓
0x06	Get scene membership	Optional	✓	✓	✓
0x40	Enhanced add scene	Optional	✓	✓	✓
0x41	Enhanced view scene	Optional	✓	✓	✓
0x42	Copy scene	Optional	✓	✓	✓

6.5.2.3.1 Generic usage notes

The use of the *add scene* command shall be enhanced in the ZLL profile to allow it to also be sent via the groupcast or broadcast transmission service.

6.6 On/off cluster

6.6.1 Server

6.6.1.1 Dependencies

6.6.1.1.1 Effect on receipt of level control cluster commands

On receipt of a *level control* cluster command that causes the *OnOff* attribute to be set to 0x00, the *OnTime* attribute shall be set to 0x0000.

On receipt of a *level control* cluster command that causes the *OnOff* attribute to be set to 0x01, if the value of the *OnTime* attribute is equal to 0x0000, the device shall set the *OffWaitTime* attribute to 0x0000.

6.6.1.2 Attributes

When a device implements the *on/off* cluster at the ZCL server side, it shall support the attributes listed in Table 27.

Table 27 – Mandatory attributes of the on/off cluster

Identifier	Name
0x0000	OnOff

In addition, the server side of the *on/off* cluster shall be enhanced with the attributes listed in Table 28.

Table 28 – Additional attributes of the server side of the on/off cluster

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x4000	GlobalScene-Control	Boolean	TRUE or FALSE	Read only	TRUE	Mandatory
0x4001	OnTime	Unsigned 16-bit integer	0x0000 – 0xffff	Read/ write	0x0000	Mandatory
0x4002	OffWaitTime	Unsigned 16-bit integer	0x0000 – 0xffff	Read/ write	0x0000	Mandatory

6.6.1.2.1 GlobalSceneControl attribute

In order to support the use case where the user gets back the last setting of the lamps, a global scene is introduced which is stored when the lamps are turned off and recalled when the lights are turned on. The global scene is defined as the scene that is stored with group identifier 0 and scene identifier 0 (see also 6.5.1.3.1).

The *GlobalSceneControl* attribute is defined in order to prevent a second *off* command storing the all-lamps-off situation as a global scene, and to prevent a second *on* command destroying the current settings by going back to the global scene.

The *GlobalSceneControl* attribute shall be set to TRUE after the reception of a standard ZCL *on* command, a ZCL *move to level (with on/off)* command, a ZCL *recall scene* command or a ZLL *on with recall global scene* command (see Section 6.6.1.4.5).

The *GlobalSceneControl* attribute is set to FALSE after reception of a ZLL *off with effect* command (See Section 6.6.1.4.1).

These concepts are illustrated in Figure 19.

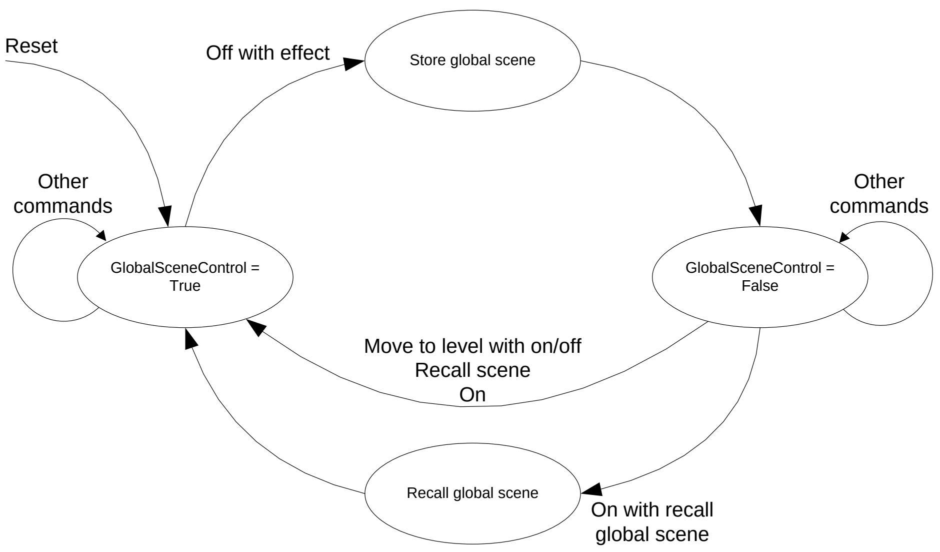

Reset
Off with effect
Store global scene
Other commands
Other commands
GlobalSceneControl = True
GlobalSceneControl = False
Move to level with on/off
Recall scene
On
Recall global scene
On with recall global scene

Figure 19 – State behavior of store and recall global scene

6.6.1.2.2 OnTime attribute

The *OnTime* attribute specifies the length of time (in 1/10ths second) that the "on" state shall be maintained before automatically transitioning to the "off" state when using the *on with timed off* command. If this attribute is set to 0x0000 or 0xffff, the lamp shall remain in its current state.

6.6.1.2.3 OffWaitTime attribute

The *OffWaitTime* attribute specifies the length of time (in 1/10ths second) that the "off" state shall be guarded to prevent an on command turning the light back to its "on" state (e.g. when leaving a room, the lights are turned off but an occupancy sensor detects the leaving person and attempts to turn the lights back on). If this attribute is set to 0x0000, the lamp shall remain in its current state.

6.6.1.3 Scene table enhancements

When a device implements the *scenes* cluster at the ZCL server side, it shall ensure the attributes listed in Table 29 are added to the scene table.

Table 29 – On/off cluster attributes to be included in the scene table

Attribute	Reference
OnOff	[R2]

6.6.1.4 Commands received

When a device implements the *on/off* cluster at the ZCL server side, it shall be able to receive the commands listed in Table 30.

Table 30 – Mandatory commands received by the server side of the on/off cluster

Identifier	Name
0x00	Off
0x01	On
0x02	Toggle

In addition, the server side of the *on/off* cluster shall be enhanced to be able to receive the commands listed in Table 31.

Table 31 – Additional commands received by the server side of the on/off cluster

Command identifier field value	Description	Mandatory/ Optional
0x40	Off with effect	Mandatory
0x41	On with recall global scene	Mandatory
0x42	On with timed off	Mandatory

6.6.1.4.1 Off command extensions

On receipt of the *off* command, the *OnTime* attribute shall be set to 0x0000.

6.6.1.4.2 On command extensions

On receipt of the *on* command, if the value of the *OnTime* attribute is equal to 0x0000, the device shall set the *OffWaitTime* attribute to 0x0000.

6.6.1.4.3 Toggle command extensions

On receipt of the *toggle* command, if the value of the *OnOff* attribute is equal to 0x00 and if the value of the *OnTime* attribute is equal to 0x0000, the device shall set the *OffWaitTime* attribute to 0x0000. If the value of the *OnOff* attribute is equal to 0x01, the *OnTime* attribute shall be set to 0x0000.

6.6.1.4.4 Off with effect command

The *off with effect* command allows lamps to be turned off using enhanced ways of fading.

The payload of this command shall be formatted as illustrated in Figure 20.

Octets	1	1
Data type	Unsigned 8-bit integer	Unsigned 8-bit integer
Field name	Effect identifier	Effect variant

Figure 20 – Format of the off with effect command

6.6.1.4.4.1 Effect identifier field

The *effect identifier* field is 8-bits in length and specifies the fading effect to use when switching the light off. This field shall contain one of the non-reserved values listed in Table 32.

Table 32 – Values of the effect identifier field of the off with effect command

***effect identifier* field value**	**Description**
0x00	Delayed all off
0x01	Dying light
0x02 – 0xff	Reserved

6.6.1.4.4.2 Effect variant field

The *effect variant* field is 8-bits in length and is used to indicate which variant of the effect, indicated in the *effect identifier* field, should be triggered. If a device does not support the given variant, it shall use the default variant. This field is dependent on the value of the *effect identifier* field and shall contain one of the non-reserved values listed in Table 33.

Table 33 – Values of the effect variant field of the off with effect command

***effect identifier* field value**	***effect variant* field value**	**Description**
0x00	0x00 (default)	Fade to off in 0.8 seconds
	0x01	No fade
	0x02	50% dim down in 0.8 seconds then fade to off in 12 seconds
	0x03 – 0xff	Reserved
0x01	0x00 (default)	20% dim up in 0.5s then fade to off in 1 second
	0x01 – 0xff	Reserved
0x02 – 0xff	0x00 – 0xff	Reserved

6.6.1.4.4.3 Effect on receipt

On receipt of the *off with effect* command and if the *GlobalSceneControl* attribute is equal to TRUE, the application on the associated endpoint shall store its settings in its global scene then set the *GlobalSceneControl* attribute to FALSE. The application shall then enter its “off” state, update the *OnOff* attribute accordingly and set the *OnTime* attribute to 0x0000.

In all other cases, the application on the associated endpoint shall enter its “off” state and update the *OnOff* attribute accordingly.

6.6.1.4.5 On with recall global scene command

The *on with recall global scene* command allows the recall of the light settings when the light was turned off.

The *on with recall global scene* command shall have no parameters.

6.6.1.4.5.1 Effect on receipt

On receipt of the *on with recall global scene* command, if the *GlobalSceneControl* attribute is equal to FALSE, the application on the associated endpoint shall recall its global scene, entering the appropriate state and updating the *OnOff* attribute accordingly. It shall then set the *GlobalSceneControl* attribute to TRUE.

In all other cases, the application on the associated endpoint shall enter its “on” state and update the *OnOff* attribute accordingly.

If the value of the *OnTime* attribute is equal to 0x0000, the device shall then set the *OffWaitTime* attribute to 0x0000.

6.6.1.4.6 On with timed off command

The *on with timed off* command allows lamps to be turned on for a specific duration with a guarded off duration so that should the lamp be subsequently switched off, further *on with timed off* commands, received during this time, are prevented from turning the lamps back on. Note that the lamp can be periodically re-kicked by subsequent *on with timed off* commands, e.g. from an on/off sensor.

The payload of this command shall be formatted as illustrated in Figure 21.

Octets	1	2	2
Data type	Unsigned 8-bit integer	Unsigned 16-bit integer	Unsigned 16-bit integer
Field name	On/off control	On time	Off wait time

Figure 21 – Format of the on with timed off command

6.6.1.4.6.1 On/off control field

The *on/off control* field is 8-bits in length and contains information on how the lamp is to be operated. This field shall be formatted as illustrated in Figure 22.

Bits: 0	**1-7**
Accept only when on	Reserved

Figure 22 – Format of the on/off control field of the on with timed off command

The *accept only when on* sub-field is 1 bit in length and specifies whether the *on with timed off* command is to be processed unconditionally or only when the *OnOff* attribute is equal to 0x01. If this sub-field is set to 1, the *on with timed off* command shall only be accepted if the *OnOff* attribute is equal to 0x01. If this sub-field is set to 0, the *on with timed off* command shall be processed unconditionally.

6.6.1.4.6.2 On time field

The *on time* field is 16 bits in length and specifies the length of time (in 1/10ths second) that the lamp is to remain "on", i.e. with its *OnOff* attribute equal to 0x01, before automatically turning "off". This field shall be specified in the range 0x0000 – 0xfffe.

6.6.1.4.6.3 Off wait time field

The *off wait time* field is 16 bits in length and specifies the length of time (in 1/10ths second) that the lamp shall remain "off", i.e. with its *OnOff* attribute equal to 0x00, and guarded to prevent an on command turning the light back "on". This field shall be specified in the range 0x0000 – 0xfffe.

6.6.1.4.6.4 Effect on receipt

On receipt of this command, if the *accept only when on* sub-field of the on/off control field is set to 1and the value of the *OnOff* attribute is equal to 0x00 (off), the command shall be discarded.

If the value of the *OffWaitTime* attribute is greater than zero and the value of the *OnOff* attribute is equal to 0x00, then the device shall set the *OffWaitTime* attribute to the minimum of the *OffWaitTime* attribute and the value specified in the off wait time field.

In all other cases, the device shall set the *OnTime* attribute to the maximum of the *OnTime* attribute and the value specified in the on time field, set the *OffWaitTime* attribute to the value specified in the off wait time field and set the *OnOff* attribute to 0x01 (on).

If the values of the *OnTime* and *OffWaitTime* attributes are both less than 0xffff, the device shall then update the device every 1/10th second until both the *OnTime* and *OffWaitTime* attributes are equal to 0x0000, as follows:

- If the value of the *OnOff* attribute is equal to 0x01 (on) and the value of the *OnTime* attribute is greater than zero, the device shall decrement the value of the *OnTime* attribute. If the value of the *OnTime* attribute reaches 0x0000, the device shall set the *OffWaitTime* and *OnOff* attributes to 0x0000 and 0x00, respectively.
- If the value of the *OnOff* attribute is equal to 0x00 (off) and the value of the *OffWaitTime* attribute is greater than zero, the device shall decrement the value of the *OffWaitTime*

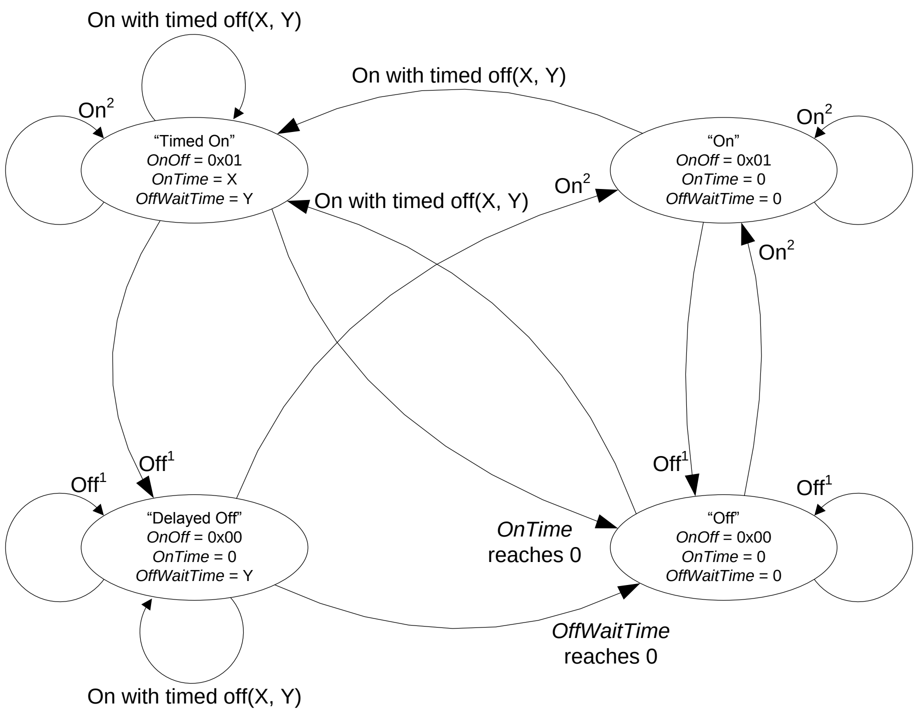
attribute. If the value of the *OffWaitTime* attribute reaches 0x0000, the device shall terminate the update.

6.6.1.5 Commands generated

No cluster specific commands are generated by the server.

6.6.1.6 State description

The operation of the on/off cluster with respect to the on, off and on with timed off commands is illustrated in Figure 23. In this diagram, the values X and Y correspond to the on time and off wait time fields, respectively, of the on with timed off command. In the "Timed On" state, the *OnTime* attribute is decremented every 1/10th second. Similarly, in the "Delayed Off" state, the *OffWaitTime* attribute is decremented every 1/10th second.



Note 1: Any command which causes the *OnOff* attribute to be set to 0x00, e.g. Off, Toggle or Off with effect.
Note 2: Any command which causes the *OnOff* attribute to be set to 0x01, e.g. On, Toogle or On with recall global scene.

Figure 23 – On/Off cluster operation state machine

6.6.2 Client

6.6.2.1 Attributes

The client has no attributes.

6.6.2.2 Commands received

No cluster specific commands are received by the client.

6.6.2.3 Commands generated

The client generates the cluster specific commands listed in Table 34, as required by the application. These commands are detailed in 6.6.1.4.

Table 34 – Commands generated by the client side of the on/off cluster

Identifier	Name	Usage	Permitted transmission services		
			Unicast	Groupcast	Broadcast
0x00	Off	Optional	✓	✓	✓
0x01	On	Optional	✓	✓	✓
0x02	Toggle	Optional	✓	✓	✓
0x40	Off with effect	Optional	✓	✓	✓
0x41	On with recall global scene	Optional	✓	✓	✓
0x42	On with timed off	Optional	✓	✓	✓

6.7 Level control cluster

6.7.1 Server

6.7.1.1 Attributes

When a device implements the level control cluster at the ZCL server side, it shall support the attributes listed in Table 35.

Table 35 – Mandatory attributes of the level control cluster

Identifier	Name
0x0000	CurrentLevel
0x0001	RemainingTime

Within this profile the *CurrentLevel* attribute shall be interpreted as follows:

- A value of 0x00 shall not be used.
- A value of 0x01 shall indicate the minimum light level that can be attained on a device.
- A value of 0xfe shall indicate the maximum light level that can be attained on a device.
- A value of 0xff shall represent an undefined value.
- All other values are application specific gradations from the minimum to the maximum light level.

6.7.1.2 Scene table enhancement

When a device implements the *level control* cluster at the ZCL server side, it shall ensure the attributes listed in Table 36 are added to the scene table.

Table 36 – Additions to the scenes table for the level control cluster

Attribute	Reference
CurrentLevel	[R2]

6.7.1.3 Commands received

When a device implements the *level control* cluster at the ZCL server side, it shall be able to receive the commands listed in Table 37.

Table 37 – Mandatory commands received by the server side of the level control cluster

Identifier	Name
0x00	Move to level
0x01	Move
0x02	Step
0x03	Stop
0x04	Move to level (with on/off)
0x05	Move (with on/off)
0x06	Step (with on/off)
0x07	Stop (with on/off)[4]

6.7.1.3.1 Generic usage notes

If a *move to level*, *move*, *step* or *stop* command is received while the device is in its off state, i.e. the *OnOff* attribute of the on/off cluster is equal to 0x00, the command shall be ignored.

6.7.1.4 Commands generated

No cluster specific commands are received by the server.

6.7.2 Client

6.7.2.1 Attributes

The client has no attributes.

6.7.2.2 Commands received

No cluster specific commands are received by the client.

6.7.2.3 Commands generated

The client generates the cluster specific commands listed in Table 38, as required by the application. These commands are detailed in 6.7.1.3.

[4] This is identical to the stop command with ID 0x03 but is refers to commands with on/off. This command was included in the ZCL for symmetry. See also [R2] .

Table 38 – Commands generated by the client side of the level control cluster

Identifier	Name	Usage	Permitted transmission services		
			Unicast	Groupcast	Broadcast
0x00	Move to level	Optional	✓	✓	✓
0x01	Move	Optional	✓	✓	✓
0x02	Step	Optional	✓	✓	✓
0x03	Stop	Optional	✓	✓	✓
0x04	Move to level (with on/off)	Optional	✓	✓	✓
0x05	Move (with on/off)	Optional	✓	✓	✓
0x06	Step (with on/off)	Optional	✓	✓	✓
0x07	Stop (with on/off)[5]	Optional	✓	✓	✓

6.8 Color Control Cluster

6.8.1 Server

6.8.1.1 Attributes

When a device implements the *color control* cluster at the ZCL server side, it shall support the attributes listed in Table 39.

[5] This is identical to the stop command with ID 0x03 but is refers to commands with on/off. This command was included in the ZCL for symmetry. See also [R2] .

Table 39 – Mandatory attributes of the color control cluster

Identifier	Name	Comments
0x0000	CurrentHue	-
0x0001	CurrentStaturation	-
0x0002	RemainingTime	-
0x0003	CurrentX	A value of zero indicates invalid and the *CurrentHue* and *CurrentSaturation* attributes are used instead to determine the color.
0x0004	CurrentY	A value of zero indicates invalid and the *CurrentHue* and *CurrentSaturation* attributes are used instead to determine the color.
0x0007	ColorTemperature	-
0x0008	ColorMode	-
0x0010	NumberOfPrimaries	This attribute is used to indicate how many primaries are supported on this device, and hence how many primary attributes listed below are defined.
0x0011	Primary1X	-
0x0012	Primary1Y	-
0x0013	Primary1Intensity	-
0x0015	Primary2X	-
0x0016	Primary2Y	-
0x0017	Primary2Intensity	-
0x0019	Primary3X	-
0x001a	Primary3Y	-
0x001b	Primary3Intensity	-
0x0020	Primary4X	-
0x0021	Primary4Y	-
0x0022	Primary4Intensity	-
0x0024	Primary5X	-
0x0025	Primary5Y	-
0x0026	Primary5Intensity	-
0x0028	Primary6X	-
0x0029	Primary6Y	-
0x002a	Primary6Intensity	-

In addition, the server side of the *color control* cluster shall be enhanced with the attributes listed in Table 40.

Table 40 – Additional attributes of the server side of the color control cluster

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x4000	EnhancedCurrentHue	Unsigned 16-bit integer	0x0000 – 0xffff	Read only	0x0000	Mandatory
0x4001	EnhancedColorMode	8-bit enumeration	0x00 – 0xff	Read only	0x00	Mandatory
0x4002	ColorLoopActive	Unsigned 8-bit integer	0x00 – 0xff	Read only	0x00	Mandatory
0x4003	ColorLoopDirection	Unsigned 8-bit integer	0x00 – 0xff	Read only	0x00	Mandatory
0x4004	ColorLoopTime	Unsigned 16-bit integer	0x0000 – 0xffff	Read only	0x0019	Mandatory
0x4005	ColorLoopStart-EnhancedHue	Unsigned 16-bit integer	0x0000 – 0xffff	Read only	0x2300	Mandatory
0x4006	ColorLoopStored-EnhancedHue	Unsigned 16-bit integer	0x0000 – 0xffff	Read only	0x0000	Mandatory
0x400a	ColorCapabilities	Unsigned 16-bit bitmap	0x0000 – 0x001f	Read only	0x0000	Mandatory
0x400b	ColorTemp-PhysicalMin	Unsigned 16-bit integer	0x0000 – 0xffff	Read only	0x0000	Mandatory
0x400c	ColorTemp-PhysicalMax	Unsigned 16-bit integer	0x0000 – 0xffff	Read only	0xffff	Mandatory

6.8.1.1.1 EnhancedCurrentHue attribute

The *EnhancedCurrentHue* attribute represents non-equidistant steps along the CIE 1931 color triangle, and it provides 16-bits precision.

The upper 8 bits of this attribute shall be used as an index in the implementation specific XY lookup table to provide the non-equidistance steps (see [R6] for an example). The lower 8 bits shall be used to interpolate between these steps in a linear way in order to provide color zoom for the user.

To provide compatibility with standard ZCL, the *CurrentHue* attribute shall contain a hue value in the range 0 to 254, calculated from the *EnhancedCurrentHue* attribute.

6.8.1.1.2 EnhancedColorMode attribute

The *EnhancedColorMode* attribute specifies which attributes are currently determining the color of the device, as detailed in Table 41.

Table 41 – Values of the EnhancedColorMode attribute

Attribute value	Attributes that determine the color
0x00	*CurrentHue* and *CurrentSaturation*
0x01	*CurrentX* and *CurrentY*
0x02	*ColorTemperature*
0x03	*EnhancedCurrentHue* and *CurrentSaturation*
0x04 – 0xff	Reserved

To provide compatibility with standard ZCL, the original *ColorMode* attribute shall indicate 'CurrentHue and CurrentSaturation' when the light uses the *EnhancedCurrentHue* attribute. If the *ColorMode* attribute is changed, e.g., due to one of the standard color control cluster commands defined in the ZCL, its new value shall be copied to the *EnhancedColorMode* attribute.

6.8.1.1.3 ColorLoopActive attribute

The *ColorLoopActive* attribute specifies the current active status of the color loop. If this attribute has the value 0x00, the color loop shall not be active. If this attribute has the value 0x01, the color loop shall be active. All other values (0x02 – 0xff) are reserved.

6.8.1.1.4 ColorLoopDirection attribute

The *ColorLoopDirection* attribute specifies the current direction of the color loop. If this attribute has the value 0x00, the *EnhancedCurrentHue* attribute shall be decremented. If this attribute has the value 0x01, the *EnhancedCurrentHue* attribute shall be incremented. All other values (0x02 – 0xff) are reserved.

6.8.1.1.5 ColorLoopTime attribute

The *ColorLoopTime* attribute specifies the number of seconds it shall take to perform a full color loop, i.e. to cycle all values of the *EnhancedCurrentHue* attribute (between 0x0000 and 0xffff).

6.8.1.1.6 ColorLoopStartEnhancedHue attribute

The *ColorLoopStartEnhancedHue* attribute specifies the value of the *EnhancedCurrentHue* attribute from which the color loop shall be started.

6.8.1.1.7 ColorLoopStoredEnhancedHue attribute

The *ColorLoopStoredEnhancedHue* attribute specifies the value of the *EnhancedCurrentHue* attribute before the color loop was started. Once the color loop is complete, the *EnhancedCurrentHue* attribute shall be restored to this value.

6.8.1.1.8 ColorCapabilities attribute

The *ColorCapabilities* attribute specifies that color capabilities of the device supporting the color control cluster, as illustrated in Table 42. If a bit is set to 1, the corresponding attributes and commands shall become mandatory. If a bit is set to 0, the corresponding attributes and commands need not be implemented.

Table 42 – Bit values of the *ColorCapabilities* attribute

ColorCapabilities bit	Description	Related attributes	Mandatory commands
0	Hue/saturation supported	*CurrentHue* *CurrentSaturation*	*Move to hue* *Move hue* *Step hue* *Move to saturation* *Move saturation* *Step saturation* *Move to hue and saturation*
1	Enhanced hue supported Note: hue/ saturation must also be supported.	*EnhancedCurrentHue*	*Enhanced move to hue* *Enhanced move hue* *Enhanced step hue* *Enhanced move to hue and saturation*
2	Color loop supported Note: enhanced hue must also be supported.	*ColorLoopActive* *ColorLoopDirection* *ColorLoopTime* *ColorLoopStartEnhancedHue* *ColorLoopStoredEnhancedHue*	*Color loop set*
3	XY attributes supported	*CurrentX* *CurrentY*	*Move to color* *Move color* *Step color*
4	Color temperature supported	*ColorTemperature* *ColorTempPhysicalMin* *ColorTempPhysicalMax*	*Move to color temperature* *Move color temperature* *Step color temperature*
5-15	Reserved	-	-

Note: The support of the *CurrentX* and *CurrentY* attributes is mandatory regardless of color capabilities.

On receipt of a unicast color control cluster command that is not supported or a general command which affects a color control cluster attribute that is not supported, the device shall respond with a default response command with a status indicating an unsupported cluster command or unsupported attribute, respectively.

6.8.1.1.9 ColorTempPhysicalMin attribute

The *ColorTempPhysicalMin* attribute indicates the minimum color temperature value supported by the hardware such that *ColorTempPhysicalMin* ≤ *ColorTemperature*.

6.8.1.1.10 ColorTempPhysicalMax attribute

The *ColorTempPhysicalMax* attribute indicates the maximum color temperature value supported by the hardware such that *ColorTemperature* ≤ *ColorTempPhysicalMax*.

6.8.1.2 Scene table enhancements

When a device implements the *scenes* cluster at the ZCL server side, it shall ensure the attributes listed in Table 43 are added to the scene table.

Table 43 – Additions to the scenes table for the color control cluster

Attribute	Reference
CurrentX	[R2]
CurrentY	[R2]
EnhancedCurrentHue	6.8.1.1.1
CurrentSaturation	[R2]
ColorLoopActive	6.8.1.1.3
ColorLoopDirection	6.8.1.1.4
ColorLoopTime	6.8.1.1.5

Since there is a direct relation between *ColorTemperature* and XY, color temperature, if supported, is stored as XY in the scenes table.

Attributes in the scene table that are not supported by the device (according to the *ColorCapabilities* attribute) shall be present but ignored.

6.8.1.3 Commands received

When a device implements the *color control* cluster at the ZCL server side, it shall be able to receive the commands listed in Table 44, depending on the value of the *ColorCapabilities* attribute (see Table 42).

Table 44 – Commands received by the server side of the color control cluster

Identifier	Name
0x00	Move to hue
0x01	Move hue
0x02	Step hue
0x03	Move to saturation
0x04	Move saturation
0x05	Step saturation
0x06	Move to hue and saturation
0x07	Move to color
0x08	Move color
0x09	Step color
0x0a	Move to color temperature

In addition, the server side of the *color control* cluster shall be enhanced to be able to receive the commands listed in Table 45.

Table 45 – Additional commands received by the server side of the color control cluster

Command identifier field value	Description
0x40	Enhanced move to hue
0x41	Enhanced move hue
0x42	Enhanced step hue
0x43	Enhanced move to hue and saturation
0x44	Color loop set
0x47	Stop move step
0x4b	Move color temperature
0x4c	Step color temperature

6.8.1.3.1 Generic usage notes

If one of these commands is received while the device is in its off state, i.e. the *OnOff* attribute of the *on/off* cluster is equal to 0x00, the command shall be ignored.

When asked to change color via one of these commands, the implementation shall select a color, within the limits of the hardware of the device, which is as close as possible to that requested. The determination as to the true representations of color is out of the scope of this specification.

If a color loop is active (i.e. the *ColorLoopActive* attribute is equal to 0x01), it shall only be stopped by sending a specific color loop set command frame with a request to deactivate the color loop (i.e. the color loop shall not be stopped on receipt of another command such as the enhanced move to hue command). In addition, while a color loop is active, a manufacturer may choose to ignore incoming color commands which affect a change in hue.

6.8.1.3.2 Enhanced move to hue command

The *enhanced move to hue* command allows lamps to be moved in a smooth continuous transition from their current hue to a target hue.

The payload of this command shall be formatted as illustrated in Figure 24.

Octets	2	1	2
Data type	Unsigned 16-bit integer	8-bit enumeration	Unsigned 16-bit integer
Field name	Enhanced hue	Direction	Transition time

Figure 24 – Format of the enhanced move to hue command

6.8.1.3.2.1 Enhanced hue field

The *enhanced hue* field is 16-bits in length and specifies the target extended hue for the lamp.

6.8.1.3.2.2 Direction field

This field is identical to the *direction* field of the *move to hue* command of the *color control* cluster (see sub-clause 5.2.2.3.2 of [R2]).

6.8.1.3.2.3 Transition time field

This field is identical to the *transition time* field of the *move to hue* command of the *color control* cluster (see sub-clause 5.2.2.3.2 of [R2]).

6.8.1.3.2.4 Effect on receipt

On receipt of this command, a device shall set the *ColorMode* attribute to 0x00 and set the *EnhancedColorMode* attribute to the value 0x03. The device shall then move from its current enhanced hue to the value given in the *enhanced hue* field.
The movement shall be continuous, i.e. not a step function, and the time taken to move to the new enhanced hue shall be equal to the *transition time* field.
Note that if the target color specified is not achievable by the hardware then the command shall not be carried out, and a ZCL *default response* command shall be generated, where not disabled, with status code equal to INVALID_VALUE.

6.8.1.3.3 Enhanced move hue command

The *enhanced move hue* command allows lamps to be moved in a continuous stepped transition from their current hue to a target hue.

The payload of this command shall be formatted as illustrated in Figure 25.

Octets	1	2
Data type	8-bit enumeration	Unsigned 16-bit integer
Field name	Move mode	Rate

Figure 25 – Format of the enhanced move hue command

6.8.1.3.3.1 Move mode field

This field is identical to the *move mode* field of the *move hue* command of the *color control* cluster (see sub-clause 5.2.2.3.3 of [R2]).

6.8.1.3.3.2 Rate field

The *rate* field is 16-bits in length and specifies the rate of movement in steps per second. A step is a change in the extended hue of a device by one unit. If the *rate* field has a value of zero, the command has no effect and a ZCL *default response* command shall be sent in response, with the status code set to INVALID_FIELD.

6.8.1.3.3.3 Effect on receipt

On receipt of this command, a device shall set the *ColorMode* attribute to 0x00 and set the *EnhancedColorMode* attribute to the value 0x03. The device shall then move from its current enhanced hue in an up or down direction in a continuous fashion, as detailed in Table 46.

Table 46 – Actions on receipt of the enhanced move hue command

Move mode	Action on receipt
Stop	If moving, stop, else ignore the command (i.e. the command is accepted but has no effect). NB This may also be used to stop an *enhanced move to hue* command or an enhanced *move to hue and saturation* command.
Up	Increase the device's enhanced hue at the rate given in the *rate* field. If the enhanced hue reaches the maximum allowed for the device, proceed to its minimum allowed value.
Down	Decrease the device's enhanced hue at the rate given in the *rate* field. If the hue reaches the minimum allowed for the device, proceed to its maximum allowed value.

6.8.1.3.4 Enhanced step hue command

The *enhanced step hue* command allows lamps to be moved in a stepped transition from their current hue to a target hue, resulting in a linear transition through XY space.

The payload of this command shall be formatted as illustrated in Figure 26.

Octets	1	2	2
Data type	8-bit enumeration	Unsigned 16-bit integer	Unsigned 16-bit integer
Field name	Step mode	Step size	Transition time

Figure 26 – Format of the enhanced step hue command

6.8.1.3.4.1 Step mode field

This field is identical to the *step mode* field of the *step hue* command of the *color control* cluster (see sub-clause 5.2.2.3.4 of [R2]).

6.8.1.3.4.2 Step size field

The *step size* field is 16-bits in length and specifies the change to be added to (or subtracted from) the current value of the device's enhanced hue.

6.8.1.3.4.3 Transition time field

The *transition time* field is 16-bits in length and specifies, in units of 1/10ths of a second, the time that shall be taken to perform the step. A step is a change to the device's enhanced hue of a magnitude corresponding to the *step size* field.

6.8.1.3.4.4 Effect on receipt

On receipt of this command, a device shall set the *ColorMode* attribute to 0x00 and the *EnhancedColorMode* attribute to the value 0x03. The device shall then move from its current enhanced hue in an up or down direction by one step, as detailed in Table 47.

Table 47 – Actions on receipt for the enhanced step hue command

Move mode	Action on receipt
Up	Increase the device’s enhanced hue by one step. If the enhanced hue reaches the maximum allowed for the device, proceed to its minimum allowed value.
Down	Decrease the device’s enhanced hue by one step. If the hue reaches the minimum allowed for the device, proceed to its maximum allowed value.

6.8.1.3.5 Enhanced move to hue and saturation command

The *enhanced move to hue and saturation* command allows lamps to be moved in a smooth continuous transition from their current hue to a target hue and from their current saturation to a target saturation.

The payload of this command shall be formatted as illustrated in Figure 27.

Octets	2	1	2
Data type	Unsigned 16-bit integer	Unsigned 8-bit integer	Unsigned 16-bit integer
Field name	Enhanced hue	Saturation	Transition time

Figure 27 – Format of the enhanced move to hue and saturation command

6.8.1.3.5.1 Enhanced hue field

The *enhanced hue* field is 16-bits in length and specifies the target extended hue for the lamp.

6.8.1.3.5.2 Saturation field

This field is identical to the *saturation* field of the *move to hue and saturation* command of the *color control* cluster (see sub-clause 5.2.2.3.8 of [R2]).

6.8.1.3.5.3 Transition time field

This field is identical to the *transition time* field of the *move to hue* command of the *color control* cluster (see sub-clause 5.2.2.3.8 of [R2]).

6.8.1.3.5.4 Effect on receipt

On receipt of this command, a device shall set the *ColorMode* attribute to the value 0x00 and set the *EnhancedColorMode* attribute to the value 0x03. The device shall then move from its current enhanced hue and saturation to the values given in the *enhanced hue* and *saturation* fields.
The movement shall be continuous, i.e. not a step function, and the time taken to move to the new color shall be equal to the *transition time* field, in 1/10ths of a second.
The path through color space taken during the transition is not specified, but it is recommended that the shortest path is taken though XY space, i.e. movement is 'in a straight line' across the hue/saturation disk. Note that if the target color specified is not achievable by the hardware then the command shall not be carried out and a ZCL default response command shall be generated, where not disabled, with status code equal to INVALID_VALUE.

6.8.1.3.6 Color loop set command

The *color loop set* command allows a color loop to be activated such that the color lamp cycles through its range of hues.

The payload of this command shall be formatted as illustrated in Figure 28.

Octets	1	1	1	2	2
Data type	8-bit bitmap	8-bit enumeration	8-bit enumeration	Unsigned 16-bit integer	Unsigned 16-bit integer
Field name	Update flags	Action	Direction	Time	Start hue

Figure 28 – Format of the color loop set command

6.8.1.3.6.1 Update flags field

The *update flags* field is 8 bits in length and specifies which color loop attributes to update before the color loop is started. This field shall be formatted as illustrated in Figure 29.

Bits: 0	**1**	**2**	**3**	**4-7**
Update action	Update direction	Update time	Update start hue	Reserved

Figure 29 – Format of the update flags field of the color loop set command

The *update action* sub-field is 1 bit in length and specifies whether the device shall adhere to the *action* field in order to process the command. If this sub-field is set to 1, the device shall adhere to the *action* field. If this sub-field is set to 0, the device shall ignore the *action* field.

The *update direction* sub-field is 1 bit in length and specifies whether the device shall update the *ColorLoopDirection* attribute with the *direction* field. If this sub-field is set to 1, the device shall update the value of the *ColorLoopDirection* attribute with the value of the *direction* field. If this sub-field is set to 0, the device shall ignore the *direction* field.

The *update time* sub-field is 1 bit in length and specifies whether the device shall update the *ColorLoopTime* attribute with the *time* field. If this sub-field is set to 1, the device shall update the value of the *ColorLoopTime* attribute with the value of the *time* field. If this sub-field is set to 0, the device shall ignore the *time* field.

The *update start hue* sub-field is 1 bit in length and specifies whether the device shall update the *ColorLoopStartEnhancedHue* attribute with the *start hue* field. If this sub-field is set to 1, the device shall update the value of the *ColorLoopStartEnhancedHue* attribute with the value of the *start hue* field. If this sub-field is set to 0, the device shall ignore the *start hue* field.

6.8.1.3.6.2 Action field

The *action* field is 8 bits in length and specifies the action to take for the color loop if the *update action* sub-field of the *update flags* field is set to 1. This field shall be set to one of the non reserved values listed in Table 48.

Table 48 – Values of the action field of the color loop set command

Action field value	Description
0x00	De-activate the color loop.
0x01	Activate the color loop from the value in the *ColorLoopStartEnhancedHue* field.
0x02	Activate the color loop from the value of the *EnhancedCurrentHue* attribute.
0x03 – 0xff	Reserved.

6.8.1.3.6.3 Direction field

Te *direction* field is 8 bits in length and specifies the direction for the color loop if the *update direction* field of the *update flags* field is set to 1. This field shall be set to one of the non reserved values listed in Table 49.

Table 49 – Values of the direction field of the color loop set command

Direction field value	Description
0x00	Decrement the hue in the color loop.
0x01	Increment the hue in the color loop.
0x02 – 0xff	Reserved.

6.8.1.3.6.4 Time field

The *time* field is 16 bits in length and specifies the number of seconds over which to perform a full color loop if the *update time* field of the *update flags* field is set to 1.

6.8.1.3.6.5 Start hue field

The *start hue* field is 16 bits in length and specifies the starting hue to use for the color loop if the *update start hue* field of the *update flags* field is set to 1.

6.8.1.3.6.6 Effect on receipt

On receipt of this command, the device shall first update its color loop attributes according to the value of the *update flags* field, as follows. If the *update direction* sub-field is set to 1, the device shall set the *ColorLoopDirection* attribute to the value of the *direction* field. If the *update time* sub-field is set to 1, the device shall set the *ColorLoopTime* attribute to the value of the *time* field. If the *update start hue* sub-field is set to 1, the device shall set the *ColorLoopStartEnhancedHue* attribute to the value of the *start hue* field. If the color loop is active (and stays active), the device shall immediately react on updates of the *ColorLoopDirection* and *ColorLoopTime* attributes.

If the *update action* sub-field of the *update flags* field is set to 1, the device shall adhere to the action specified in the *action* field, as follows. If the value of the *action* field is set to 0x00, the device shall de-active the color loop, set the *ColorLoopActive* attribute to 0x00 and set the *EnhancedCurrentHue* attribute to the value of the *ColorLoopStoredEnhancedHue* attribute. If the value of the *action* field is set to 0x01, the device shall set the *ColorLoopStoredEnhancedHue* attribute to the value of the *EnhancedCurrentHue* attribute, set the *ColorLoopActive* attribute to 0x01 and activate the color loop, starting from the value of the *ColorLoopStartEnhancedHue* attribute. If the value of the *action* field is set to 0x02, the device shall set the *ColorLoopStoredEnhancedHue* attribute to the value of the *EnhancedCurrentHue* attribute, set the *ColorLoopActive* attribute to 0x01 and activate the color loop, starting from the value of the *EnhancedCurrentHue* attribute.

If the color loop is active, the device shall cycle over the complete range of values of the *EnhancedCurrentHue* attribute in the direction of the *ColorLoopDirection* attribute over the time specified in the *ColorLoopTime* attribute. The level of increments/decrements is application specific.

6.8.1.3.7 Stop move step command

In the *color control* cluster, an explicit stop command is not available. Instead, it is only available as an option for the *move* commands. The *stop move step* command is provided to allow *move to* and *step* commands to be stopped. (Note this automatically provides symmetry to the *level control* cluster.)

The *stop move step* command shall have no payload.

Note: the *stop move step* command has no effect on an active color loop.

6.8.1.3.7.1 Effect on receipt

Upon receipt of this command, any *move to*, *move* or *step* command currently in process shall be terminated. The values of the *CurrentHue*, *EnhancedCurrentHue* and *CurrentSaturation* attributes shall be left at their present value upon receipt of the *stop move step* command, and the *RemainingTime* attribute shall be set to zero.

6.8.1.3.8 Move color temperature command

The *move color temperature* command allows the color temperature of a lamp to be moved at a specified rate.

The payload of this command shall be formatted as illustrated in Figure 30.

Octets	1	2	2	2
Data type	8-bit bitmap	Unsigned 16-bit integer	Unsigned 16-bit integer	Unsigned 16-bit integer
Field name	Move mode	Rate	Color temperature minimum	Color temperature maximum

Figure 30 – Format of the move color temperature command

6.8.1.3.8.1 Move mode field

This field is identical to the *move mode* field of the *move hue* command of the *color control* cluster (see sub-clause 5.2.2.3.3 of [R2]).

6.8.1.3.8.2 Rate field

The *rate* field is 16-bits in length and specifies the rate of movement in steps per second. A step is a change in the color temperature of a device by one unit. If the *rate* field has a value of zero, the command has no effect and a ZCL *default response* command shall be sent in response, with the status code set to INVALID_FIELD.

6.8.1.3.8.3 Color temperature minimum field

The *color temperature minimum* field is 16-bits in length and specifies a lower bound on the color temperature for the current move operation such that:

$$ColorTempPhysicalMin \leq color\ temperature\ minimum \leq ColorTemperature$$

As such if the move operation takes the *ColorTemperature* attribute towards *color temperature minimum* it shall be clipped so that the above invariant is satisfied. If the *color temperature minimum* field is set to 0x0000, *ColorTempPhysicalMin* shall be used as the lower bound for the color temperature.

6.8.1.3.8.4 Color temperature maximum field

The *color temperature maximum* field is 16-bits in length and specifies an upper bound on the color temperature for the current move operation such that:

$$ColorTemperature \leq color\ temperature\ maximum \leq ColorTempPhysicalMax$$

As such if the move operation takes the *ColorTemperature* attribute towards *color temperature maximum* it shall be clipped so that the above invariant is satisfied. If the *color temperature maximum* field is set to 0x0000, *ColorTempPhysicalMax* shall be used as the upper bound for the color temperature.

6.8.1.3.8.5 Effect on receipt

On receipt of this command, a device shall set both the *ColorMode* and *EnhancedColorMode* attributes to 0x02. The device shall then move from its current color temperature in an up or down direction in a continuous fashion, as detailed in Table 50.

Table 50 – Actions on receipt of the move color temperature command

Move mode	Action on receipt
Stop	If moving, stop the operation, else ignore the command (i.e. the command is accepted but has no effect).
Up	Increase the device's color temperature at the rate given in the *rate* field. If the color temperature reaches the maximum allowed for the device (via either the *color temperature maximum* field or the *ColorTempPhysicalMax* attribute), the move operation shall be stopped.
Down	Decrease the device's color temperature at the rate given in the *rate* field. If the color temperature reaches the minimum allowed for the device (via either the *color temperature minimum* field or the *ColorTempPhysicalMin* attribute), the move operation shall be stopped.

6.8.1.3.9 Step color temperature command

The *step color temperature* command allows the color temperature of a lamp to be stepped with a specified step size.

The payload of this command shall be formatted as illustrated in Figure 31.

Octets	1	2	2	2	2
Data type	8-bit bitmap	Unsigned 16-bit integer	Unsigned 16-bit integer	Unsigned 16-bit integer	Unsigned 16-bit integer
Field name	Step mode	Step size	Transition time	Color temperature minimum	Color temperature maximum

Figure 31 – Format of the step color temperature command

6.8.1.3.9.1 Step mode field

This field is identical to the *step mode* field of the *step hue* command of the *color control* cluster (see sub-clause 5.2.2.3.4 of [R2]).

6.8.1.3.9.2 Step size field

The *step size* field is 16-bits in length and specifies the change to be added to (or subtracted from) the current value of the device's color temperature.

6.8.1.3.9.3 Transition time field

The *transition time* field is 16-bits in length and specifies, in units of 1/10ths of a second, the time that shall be taken to perform the step. A step is a change to the device's color temperature of a magnitude corresponding to the *step size* field.

6.8.1.3.9.4 Color temperature minimum field

The *color temperature minimum* field is 16-bits in length and specifies a lower bound on the color temperature for the current step operation such that:

$$\mathit{ColorTempPhysicalMin} \le \textit{color temperature minimum} \le \mathit{ColorTemperature}$$

As such if the step operation takes the *ColorTemperature* attribute towards *color temperature minimum* it shall be clipped so that the above invariant is satisfied. If the *color temperature minimum* field is set to 0x0000, *ColorTempPhysicalMin* shall be used as the lower bound for the color temperature.

6.8.1.3.9.5 Color temperature maximum field

The *color temperature maximum* field is 16-bits in length and specifies an upper bound on the color temperature for the current step operation such that:

$$\mathit{ColorTemperature} \le \textit{color temperature maximum} \le \mathit{ColorTempPhysicalMax}$$

As such if the step operation takes the *ColorTemperature* attribute towards *color temperature maximum* it shall be clipped so that the above invariant is satisfied. If the *color temperature maximum* field is set to 0x0000, *ColorTempPhysicalMax* shall be used as the upper bound for the color temperature.

6.8.1.3.9.6 Effect on receipt

On receipt of this command, a device shall set both the *ColorMode* and *EnhancedColorMode* attributes to 0x02. The device shall then move from its current color temperature in an up or down direction by one step, as detailed in Table 51.

Table 51 – Actions on receipt for the step color temperature command

Move mode	Action on receipt
Up	Increase the device’s color temperature by one step. If the color temperature reaches the maximum allowed for the device (via either the *color temperature maximum* field or the *ColorTempPhysicalMax* attribute), the step operation shall be stopped.
Down	Decrease the device’s color temperature by one step. If the color temperature reaches the minimum allowed for the device (via either the *color temperature minimum* field or the *ColorTempPhysicalMin* attribute), the step operation shall be stopped.

6.8.1.4 Commands generated

No cluster specific commands are generated by the server.

6.8.2 Client

6.8.2.1 Attributes

The client has no attributes.

6.8.2.2 Commands received

No cluster specific commands are received by the client.

6.8.2.3 Commands generated

The client generates the cluster specific commands listed in Table 52, as required by the application. These commands are detailed in 6.8.1.3.

Table 52 – Commands generated by the client side of the color control cluster

Identifier	**Name**	**Usage**	**Permitted transmission services**		
			Unicast	**Groupcast**	**Broadcast**
0x00	Move to hue	Optional	✓	✓	✓
0x01	Move hue	Optional	✓	✓	✓
0x02	Step hue	Optional	✓	✓	✓
0x03	Move to saturation	Optional	✓	✓	✓
0x04	Move saturation	Optional	✓	✓	✓
0x05	Step saturation	Optional	✓	✓	✓
0x06	Move to hue and saturation	Optional	✓	✓	✓
0x07	Move to color	Optional	✓	✓	✓
0x08	Move color	Optional	✓	✓	✓
0x09	Step color	Optional	✓	✓	✓
0x0a	Move to color temperature	Optional	✓	✓	✓
0x40	Enhanced move to hue	Optional	✓	✓	✓
0x41	Enhanced move hue	Optional	✓	✓	✓
0x42	Enhanced step hue	Optional	✓	✓	✓
0x43	Enhanced move to hue and saturation	Optional	✓	✓	✓
0x44	Color loop set	Optional	✓	✓	✓
0x47	Stop move step	Optional	✓	✓	✓
0x4b	Move color temperature	Optional	✓	✓	✓
0x4c	Step color temperature	Optional	✓	✓	✓

7 New clusters

7.1 ZLL commissioning cluster

The *ZLL commissioning* cluster provides commands to support touchlink commissioning. This cluster should not be considered part of a sub-device but rather part of the entire device. The ZLL commissioning cluster is comprised of two sets of commands – one providing touchlink commissioning functionality and one providing commissioning utility functionality.

The touchlink commissioning command set has command identifiers in the range 0x00 – 0x3f and shall be transmitted using the inter-PAN transmission service (see 8.1.10).

The commissioning utility command set has command identifiers in the range 0x40 – 0xff and shall be transmitted using the standard unicast transmission service, similar to that used for other ZCL cluster commands. These commands enable the exchange of control information between controllers (i.e., devices with a device identifier in the range 0x0800 – 0x0850).

A controller application endpoint may send an *endpoint information* command frame to another controller application endpoint to announce itself. It is then up to the recipient controller application endpoint to decide to take further action to get information about the lights that are controlled by the originator. If it decides to do so, it can use the *get group identifiers request* command frame to get knowledge about the group of lights controlled by the originator. The originator responds with a *get group identifiers response* command frame containing the requested information (which may have a start index field and a count field equal to 0, indicating no groups are used). Similarly, the recipient device can use the *get endpoint list request* command frame to get knowledge about the list of individual lights controlled by the originator. The originator responds with a *get endpoint list response* command frame containing the requested information (which may have a start index field and a count field equal to 0, indicating no lights are controlled).

Note: A typical ZLL controller application will likely reside inside battery powered remote controllers on top of a ZigBee sleeping end-device. As such, care should be taken as to when to send these commands to ensure the recipient is awake. It is recommended that such commands are sent just after touchlink commissioning between two controllers when the devices are not yet asleep and still polling for data from their parent.

7.1.1 Overview

The *ZLL commissioning* cluster shall have a cluster identifier of 0x1000. Those commands in the touchlink commissioning command set shall be sent using the ZLL profile identifier, 0xc05e whereas those commands in the commissioning utility command set shall sent using the ZHA profile identifier, 0x0104.

7.1.2 Server

7.1.2.1 Attributes

The server has no attributes.

7.1.2.2 Commands received

When a device implements the *ZLL commissioning* cluster at the ZCL server side, it shall be able to receive the commands listed in Table 53.

Table 53 – Commands received by the server side of the ZLL commissioning cluster

	Command identifier field value	Description	Mandatory/ optional	Reference
Touchlink	0x00	Scan request	Mandatory	7.1.2.2.1
	0x02	Device information request	Mandatory	7.1.2.2.2
	0x06	Identify request	Mandatory	7.1.2.2.3
	0x07	Reset to factory new request	Mandatory	7.1.2.2.4
	0x10	Network start request	Mandatory	7.1.2.2.5
	0x12	Network join router request	Mandatory	7.1.2.2.6
	0x14	Network join end device request	Mandatory	7.1.2.2.7
	0x16	Network update request	Mandatory	7.1.2.2.8
	All other values in the range 0x00 – 0x3f	Reserved	-	-
Utility	0x41	Get group identifiers request	Mandatory[6]	7.1.2.2.9
	0x42	Get endpoint list request	Mandatory[6]	7.1.2.2.10
	All other values in the range 0x40 – 0xff	Reserved	-	-

7.1.2.2.1 Scan request command frame

The *scan request* command frame is used to initiate a scan for other devices in the vicinity of the originator. The information contained in this command frame relates to the scan request initiator.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 0 (no ACK requested) and 0b10 (short network address), respectively, the destination address field shall be set to 0xffff (broadcast network address) and the source PAN ID field shall be set to any value in the range 0x0001 – 0xfffe, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b10 (broadcast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the command identifier shall be set to 0x00 (scan request).

The ZCL payload field shall contain the *scan request* command frame itself, formatted as illustrated in Figure 32.

[6] These commands are mandatory only for controller devices, i.e. those with device identifiers 0x0800, 0x0810, 0x0820, 0x0830, 0x0840 and 0x0850.

Field name	Data type	Octets
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
ZigBee information	8-bit bitmap	1
ZLL information	8-bit bitmap	1

Figure 32 – Format of the scan request command frame

7.1.2.2.1.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This field shall contain a 32-bit non-zero random number and is used to identify the current transaction (see also 8.1.11).

7.1.2.2.1.2 ZigBee information field

The *ZigBee information* field is 8-bits in length and specifies information related to ZigBee. This field shall be formatted as illustrated in Figure 33.

Bits: 0-1	2	3-7
Logical type	Rx on when idle	Reserved

Figure 33 – Format of the ZigBee information field

The *logical type* subfield is 2-bits in length and specifies the logical type of the device. The value of this subfield shall be set to 0b00 for a coordinator, 0b01 for a router or 0b10 for an end device.

The *Rx on when idle* subfield is 1 bit in length and specifies the *RxOnWhenIdle* state of the device. The value of this subfield shall be set to 1 to indicate that the receiver is left on when the device is idle or 0 otherwise.

7.1.2.2.1.3 ZLL information field

The *ZLL information* field is 8-bits in length and specifies ZLL specific information. This field shall be formatted as illustrated in Figure 34.

Bits: 0	1	2-3	4	5	6-7
Factory new	Address assignment	Reserved	Link initiator	Undefined (can be 0 or 1)	Reserved

Figure 34 – Format of the scan request ZLL information field

The *factory new* subfield is 1 bit in length and specifies whether the device is factory new. The value of this subfield shall be set to 1 to indicate the device is factory new or 0 otherwise.

The *address assignment* subfield is 1 bit in length and specifies whether the device is capable of assigning addresses. The value of this subfield shall be set to 1 to indicate the device is capable of assigning addresses or 0 otherwise.

The *link initiator* subfield is 1 bit in length and specifies whether the device is capable of initiating a link operation. The value of this subfield shall be set to 1 to indicate the device is capable of initiating a link (i.e. it supports the ZLL commissioning cluster at the client side) or 0 otherwise (i.e. it does not support the ZLL commissioning cluster at the client side).

7.1.2.2.2 Device information request command frame

The *device information request* command frame is used to request information regarding the sub-devices of a remote device.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in the preceding *scan request* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the command identifier shall be set to 0x02 (device information request).

The ZCL payload field shall contain the *device information request* command frame itself, formatted as illustrated in Figure 35.

Field name	**Data type**	**Octets**
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
Start index	Unsigned 8-bit integer	1

Figure 35 – Format of the device information request command frame

7.1.2.2.2.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN command frame sent by the initiator.

7.1.2.2.2.2 Start index field

The *start index* field is 8-bits in length and specifies the starting index (starting from 0) into the device table from which to get device information.

7.1.2.2.3 Identify request command frame

The *identify request* command frame is used to request that the recipient identifies itself in some application specific way to aid with touchlinking.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in the preceding *scan request* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the command identifier shall be set to 0x06 (identify request).

The ZCL payload field shall contain the *identify request* command frame itself, formatted as illustrated in Figure 36.

Field name	Data type	Octets
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
Identify duration	Unsigned 16-bit integer	2

Figure 36 – Format of the identify request command frame

7.1.2.2.3.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN command frame sent by the initiator.

7.1.2.2.3.2 Identify duration field

The *identify duration* field is 16-bits in length and shall specify the length of time the recipient is to remain in identify mode. The value of this field shall be set to one of the values listed in Table 54.

Table 54 – Values of the identify duration field

***Identify duration* field value**	**Description**
0x0000	Exit identify mode.
0x0001 – 0xfffe	Number of seconds to remain in identify mode.
0xffff	Remain in identify mode for a default time known by the receiver.

Note that if a device is not capable of identifying for the exact time specified in the identify duration field, it shall identify itself for a duration as close as possible to the requested value.

7.1.2.2.4 Reset to factory new request command frame

The *reset to factory new request* command frame is used to request that the recipient resets itself back to its factory new state.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in the preceding *scan request* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the command identifier shall be set to 0x07 (reset to factory new request).

The ZCL payload field shall contain the *reset to factory new request* command frame itself and this shall be formatted as illustrated in Figure 37.

Field name	Data type	Octets
Inter-PAN transaction identifier	Unsigned 32-bit integer	4

Figure 37 – Format of the reset to factory new request command frame

7.1.2.2.4.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This field shall contain a non-zero 32-bit random number and is used to identify the current reset to factory new request.

7.1.2.2.5 Network start request command frame

The *network start request* command frame is used by a factory new initiator to form a new network with a router.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in the preceding *scan request* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the command identifier shall be set to 0x10 (network start request).

The ZCL payload field shall contain the *network start request* command frame itself, formatted as illustrated in Figure 38.

Field name	**Data type**	**Octets**
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
Extended PAN identifier	IEEE address	8
Key index	Unsigned 8-bit integer	1
Encrypted network key	128-bit security key	16
Logical channel	Unsigned 8-bit integer	1
PAN identifier	Unsigned 16-bit integer	2
Network address	Unsigned 16-bit integer	2
Group identifiers begin	Unsigned 16-bit integer	2
Group identifiers end	Unsigned 16-bit integer	2
Free network address range begin	Unsigned 16-bit integer	2
Free network address range end	Unsigned 16-bit integer	2
Free group identifier range begin	Unsigned 16-bit integer	2
Free group identifier range end	Unsigned 16-bit integer	2
Initiator IEEE address	IEEE address	8
Initiator network address	Unsigned 16-bit integer	2

Figure 38 – Format of the network start request command frame

7.1.2.2.5.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN command frame sent by the initiator.

7.1.2.2.5.2 Extended PAN identifier field

The *extended PAN identifier* field is 64-bits in length and shall contain the extended PAN identifier of the new network. If this value is equal to zero, the target shall determine the extended PAN identifier for the new network.

7.1.2.2.5.3 Key index field

The *key index* field is 8-bits in length and shall specify the index (in the range 0x00 – 0x0f) of the key (and hence the protection method) to be used in the *encrypted network key* field (see also Table 67).

7.1.2.2.5.4 Encrypted network key field

The *encrypted network key* field is 128-bits in length and shall specify the network key to use for the network (see 8.7), encrypted according to the algorithm indicated by the *key index* field.

7.1.2.2.5.5 Logical channel field

The *logical channel* field is 8-bits in length and shall contain the ZLL channel to be used for the new network. If this value is equal to zero, the target shall determine the logical channel for the new network.

7.1.2.2.5.6 PAN identifier field

The *PAN identifier* field is 16-bits in length and shall contain the identifier of the new PAN. If this value is equal to zero, the target shall determine the PAN identifier for the new network.

7.1.2.2.5.7 Network address field

The *network address* field is 16-bits in length and contains the short network address (in the range 0x0001 – 0xfff7) assigned to the recipient.

7.1.2.2.5.8 Group identifiers begin field

The *group identifiers begin* field is 16-bits in length and specifies the start of the range of group identifiers that the recipient can use for its endpoints. If this value is equal to zero, a range of group identifiers has not been allocated.

7.1.2.2.5.9 Group identifiers end field

The *group identifiers end* field is 16-bits in length and specifies the end of the range of group identifiers that the recipient can use for its endpoints. If the value of the *group identifiers begin* field is equal to zero, a range of group identifiers has not been allocated and this field shall be ignored.

7.1.2.2.5.10 Free network address range begin field

The *free network address range begin* field is 16-bits in length and shall contain the value of the *aplFreeNwkAddrRangeBegin* attribute, specifying the start of the range of network addresses that the recipient can assign. If this value is equal to zero, a range of network addresses has not been allocated by the initiator for subsequent allocation by the target.

7.1.2.2.5.11 Free network address range end field

The *free network address range end* field is 16-bits in length and shall contain the value of the *aplFreeNwkAddrRangeEnd* attribute, specifying the end of the range of network addresses that the recipient can assign. If the value of the *free network address range begin* field is equal to zero, a range of network addresses has not been allocated by the initiator for subsequent allocation by the target and this field shall be ignored.

7.1.2.2.5.12 Free group identifier range begin field

The *free group identifiers begin* field is 16-bits in length and shall contain the value of the *aplFreeGroupIDRangeBegin* attribute, specifying the start of the range of group identifiers that the recipient can assign. If this value is equal to zero, a range of group identifiers has not been allocated by the initiator for subsequent allocation by the target.

7.1.2.2.5.13 Free group identifier range end field

The *free group identifiers end* field is 16-bits in length and shall contain the value of the *aplFreeGroupIDRangeEnd* attribute, specifying the end of the range of group identifiers that the recipient can assign. If the value of the *free group identifier range begin* field is equal to zero, a range of group identifiers has not been allocated by the initiator for subsequent allocation by the target and this field shall be ignored.

7.1.2.2.5.14 Initiator IEEE address

The *initiator IEEE address* is 64-bits in length and shall contain the IEEE address of the initiator of the new network.

7.1.2.2.5.15 Initiator network address field

The *initiator network address* is 16-bits in length and shall contain the short network address of the initiator of the new network.

7.1.2.2.6 Network join router request command frame

The *network join router* request command frame is used by a non factory new initiator to join a router to its network.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in the preceding *scan request* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the command identifier shall be set to 0x12 (network join router request).

The ZCL payload field shall contain the *network join router* request command frame itself, formatted as illustrated in Figure 39.

Field name	Data type	Octets
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
Extended PAN identifier	IEEE address	8
Key index	Unsigned 8-bit integer	1
Encrypted network key	128-bit security key	16
Network update identifier	Unsigned 8-bit integer	1
Logical channel	Unsigned 8-bit integer	1
PAN identifier	Unsigned 16-bit integer	2
Network address	Unsigned 16-bit integer	2
Group identifiers begin	Unsigned 16-bit integer	2
Group identifiers end	Unsigned 16-bit integer	2
Free network address range begin	Unsigned 16-bit integer	2
Free network address range end	Unsigned 16-bit integer	2
Free group identifier range begin	Unsigned 16-bit integer	2
Free group identifier range end	Unsigned 16-bit integer	2

Figure 39 – Format of the network join router request command frame

7.1.2.2.6.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN command frame sent by the initiator.

7.1.2.2.6.2 Extended PAN identifier field

The *extended PAN identifier* field is 64-bits in length and shall contain the extended PAN identifier of the network.

7.1.2.2.6.3 Key index field

The *key index* field is 8-bits in length and shall specify the index (in the range 0x00 – 0x0f) of the key (and hence the protection method) to be used in the *encrypted network key* field (see also Table 67).

7.1.2.2.6.4 Encrypted network key field

The *encrypted network key* field is 128-bits in length and shall specify the network key to use for the network (see 8.7), encrypted according to the algorithm indicated by the *key index* field.

7.1.2.2.6.5 Network update identifier field

The *network update identifier* field is 8-bits in length and shall specify the value of the *nwkUpdateId* attribute of the initiator.

7.1.2.2.6.6 Logical channel field

The *logical channel* field is 8-bits in length and shall contain the ZLL channel to be used for the network.

7.1.2.2.6.7 PAN identifier field

The *PAN identifier* field is 16-bits in length and shall contain the PAN identifier of the network.

7.1.2.2.6.8 Network address field

The *network address* field is 16-bits in length and contains the short network address assigned to the target.

7.1.2.2.6.9 Group identifiers begin field

The *group identifiers begin* field is 16-bits in length and specifies the start of the range of group identifiers that the router can use for its endpoints. If this value is equal to zero, a range of group identifiers has not been allocated.

7.1.2.2.6.10 Group identifiers end field

The *group identifiers end* field is 16-bits in length and specifies the end of the range of group identifiers that the router can use for its endpoints. If the value of the *group identifiers begin* field is equal to zero, a range of group identifiers has not been allocated and this field shall be ignored.

7.1.2.2.6.11 Free network address range begin field

The *free network address range begin* field is 16-bits in length and shall contain the value of the *aplFreeNwkAddrRangeBegin* attribute, specifying the start of the range of network addresses that the router can assign. If this value is equal to zero, a range of network addresses has not been allocated by the initiator for subsequent allocation by the target.

7.1.2.2.6.12 Free network address range end field

The *free network address range end* field is 16-bits in length and shall contain the value of the *aplFreeNwkAddrRangeEnd* attribute, specifying the end of the range of network addresses that the router can assign. If the value of the *free network address range begin* field is equal to zero, a range of network addresses has not been allocated by the initiator for subsequent allocation by the target and this field shall be ignored.

7.1.2.2.6.13 Free group identifier range begin field

The *free group identifiers begin* field is 16-bits in length and shall contain the value of the *aplFreeGroupIDRangeBegin* attribute, specifying the start of the range of group identifiers that the router can assign. If this value is equal to zero, a range of group identifiers has not been allocated by the initiator for subsequent allocation by the target.

7.1.2.2.6.14 Free group identifier range end field

The *free group identifiers end* field is 16-bits in length and shall contain the value of the *aplFreeGroupIDRangeEnd* attribute, specifying the end of the range of group identifiers that the router can assign. If the value of the *free group identifier range begin* field is equal to zero, a range of group identifiers has not been allocated by the initiator for subsequent allocation by the target and this field shall be ignored.

7.1.2.2.7 Network join end device request command frame

The *network join end device request* command frame is used by a non factory new initiator to join a factory new end device to its network.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in the preceding *scan request* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the command identifier shall be set to 0x14 (network join end device request).

The ZCL payload field shall contain the *network join end device request* command frame itself, formatted as illustrated in Figure 40.

Field name	**Data type**	**Octets**
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
Extended PAN identifier	IEEE address	8
Key index	Unsigned 8-bit integer	1
Encrypted network key	128-bit security key	16
Network update identifier	Unsigned 8-bit integer	1
Logical channel	Unsigned 8-bit integer	1
PAN identifier	Unsigned 16-bit integer	2
Network address	Unsigned 16-bit integer	2
Group identifiers begin	Unsigned 16-bit integer	2
Group identifiers end	Unsigned 16-bit integer	2
Free network address range begin	Unsigned 16-bit integer	2
Free network address range end	Unsigned 16-bit integer	2
Free group identifier range begin	Unsigned 16-bit integer	2
Free group identifier range end	Unsigned 16-bit integer	2

Figure 40 – Format of the network join end device request command frame

7.1.2.2.7.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN command frame sent by the initiator.

7.1.2.2.7.2 Extended PAN identifier field

The *extended PAN identifier* field is 64-bits in length and shall contain the extended PAN identifier of the network.

7.1.2.2.7.3 Key index field

The *key index* field is 8-bits in length and shall specify the index (in the range 0x00 – 0x0f) of the key (and hence the protection method) to be used in the *encrypted network key* field (see also Table 67).

7.1.2.2.7.4 Encrypted network key field

The *encrypted network key* field is 128-bits in length and shall specify the network key to use for the network (see 8.7), encrypted according to the algorithm indicated by the *key index* field.

7.1.2.2.7.5 Network update identifier field

The *network update identifier* field is 8-bits in length and shall specify the current value of the *nwkUpdateId* attribute of the originator.

7.1.2.2.7.6 Logical channel field

The *logical channel* field is 8-bits in length and shall contain the ZLL channel to be used for the network.

7.1.2.2.7.7 PAN identifier field

The *PAN identifier* field is 16-bits in length and shall contain the PAN identifier of the network.

7.1.2.2.7.8 Network address field

The *network address* field is 16-bits in length and contains the short network address assigned to the target.

7.1.2.2.7.9 Group identifiers begin field

The *group identifiers begin* field is 16-bits in length and specifies the start of the range of group identifiers that the end device can use for its endpoints. If this value is equal to zero, a range of group identifiers has not been allocated.

7.1.2.2.7.10 Group identifiers end field

The *group identifiers end* field is 16-bits in length and specifies the end of the range of group identifiers that the end device can use for its endpoints. If the value of the *group identifiers begin* field is equal to zero, a range of group identifiers has not been allocated and this field shall be ignored.

7.1.2.2.7.11 Free network address range begin field

The *free network address range begin* field is 16-bits in length and shall contain the value of the *aplFreeNwkAddrRangeBegin* attribute, specifying the start of the range of network addresses that the end device can assign. If this value is equal to zero, a range of network addresses has not been allocated by the initiator for subsequent allocation by the target.

7.1.2.2.7.12 Free network address range end field

The *free network address range end* field is 16-bits in length and shall contain the value of the *aplFreeNwkAddrRangeEnd* attribute, specifying the end of the range of network addresses that the end device can assign. If the value of the *free network address range begin* field is equal to zero, a range of network addresses has not been allocated by the initiator for subsequent allocation by the target and this field shall be ignored.

7.1.2.2.7.13 Free group identifier range begin field

The *free group identifiers begin* field is 16-bits in length and shall contain the value of the *aplFreeGroupIDRangeBegin* attribute, specifying the start of the range of group identifiers that the end device can assign. If this value is equal to zero, a range of group identifiers has not been allocated by the initiator for subsequent allocation by the target.

7.1.2.2.7.14 Free group identifier range end field

The *free group identifiers end* field is 16-bits in length and shall contain the value of the *aplFreeGroupIDRangeEnd* attribute, specifying the end of the range of group identifiers that the end device can assign. If the value of the *free group identifier range begin* field is equal to zero, a range of group identifiers has not been allocated by the initiator for subsequent allocation by the target and this field shall be ignored.

7.1.2.2.8 Network update request command frame

The *network update request* command frame is used to attempt to bring a router that may have missed a network update back onto the network.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the PAN identifier of the initiating device. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the command identifier shall be set to 0x16 (network update request).

The ZCL payload field shall contain the *network update request* command frame itself, formatted as illustrated in Figure 41.

Field name	**Data type**	**Octets**
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
Extended PAN identifier	IEEE address	8
Network update identifier	Unsigned 8-bit integer	1
Logical channel	Unsigned 8-bit integer	1
PAN identifier	Unsigned 16-bit integer	2
Network address	Unsigned 16-bit integer	2

Figure 41 – Format of the network update request command frame

7.1.2.2.8.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This field shall contain a non-zero 32-bit random number and is used to identify the current network update request.

7.1.2.2.8.2 Extended PAN identifier field

The *extended PAN identifier* field is 64-bits in length and shall contain the extended PAN identifier of the network.

7.1.2.2.8.3 Network update identifier field

The *network update identifier* field is 8-bits in length and shall specify the current value of the *nwkUpdateId* attribute of the originator.

7.1.2.2.8.4 Logical channel field

The *logical channel* field is 8-bits in length and shall contain the ZLL channel to be used for the network.

7.1.2.2.8.5 PAN identifier field

The *PAN identifier* field is 16-bits in length and shall contain the PAN identifier of the network.

7.1.2.2.8.6 Network address field

The *network address* field is 16-bits in length and contains the short network address assigned to the target.

7.1.2.2.9 Get group identifiers request command

The *get group identifiers request* command is used to retrieve the actual group identifiers that the endpoint is using in its multicast communication in controlling different (remote) devices.

This command shall be formatted as illustrated in Figure 42.

Octets	1
Data type	Unsigned 8-bit integer
Field name	Start index

Figure 42 – Format of the get group identifiers request command

7.1.2.2.9.1 Start index field

The *start index* field is 8-bits in length and shall contain the index (starting from 0) at which to start returning group identifiers.

7.1.2.2.10 Get endpoint list request command

The *get endpoint list request* command is used to retrieve addressing information for each endpoint the device is using in its unicast communication in controlling different (remote) devices.

This command shall be formatted as illustrated in Figure 43.

Octets	1
Data type	Unsigned 8-bit integer
Field name	Start index

Figure 43 – Format of the get endpoint list request command

7.1.2.2.10.1 Start index field

The *start index* field is 8-bits in length and shall contain the index (starting from 0) at which to start returning endpoint identifiers.

7.1.2.3 Commands generated

When a device implements the *ZLL commissioning* cluster at the ZCL server side, it shall be able to generate the commands listed in Table 55.

Table 55 – Commands generated by the server side of the ZLL commissioning cluster

	Command identifier field value	Description	Mandatory/ optional	Reference
Touchlink	0x01	Scan response	Mandatory	7.1.2.3.1
	0x03	Device information response	Mandatory	7.1.2.3.2
	0x11	Network start response	Mandatory	7.1.2.3.3
	0x13	Network join router response	Mandatory	7.1.2.3.4
	0x15	Network join end device response	Mandatory	7.1.2.3.5
	All other values in the range 0x00 – 0x3f	Reserved	-	-
Utility	0x40	Endpoint information	Mandatory[7]	7.1.2.3.6
	0x41	Get group identifiers response	Mandatory[7]	7.1.2.3.7
	0x42	Get endpoint list response	Mandatory[7]	7.1.2.3.8
	All other values in the range 0x40 – 0xff	Reserved	-	-

7.1.2.3.1 Scan response command frame

The *scan response* command frame is used to respond to the originator of a *scan request* command frame with device details. The information contained in this command frame relates to the target that is responding to the scan request command frame.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to any value in the range 0x0001 – 0xfffe, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 1 (server to client) and the command identifier shall be set to 0x01 (scan response).

The ZCL payload field shall contain the *scan response* command frame itself, formatted as illustrated in Figure 44.

[7] These commands are mandatory only for controller devices, i.e. those with device identifiers 0x0800, 0x0810, 0x0820, 0x0830, 0x0840 and 0x0850.

Field name	**Data type**	**Octets**
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
RSSI correction	Unsigned 8-bit integer	1
ZigBee information	8-bit bitmap	1
ZLL information	8-bit bitmap	1
Key bitmask	16-bit bitmap	2
Response identifier	Unsigned 32-bit integer	4
Extended PAN identifier	IEEE address	8
Network update identifier	Unsigned 8-bit integer	1
Logical channel	Unsigned 8-bit integer	1
PAN identifier	Unsigned 16-bit integer	2
Network address	Unsigned 16-bit integer	2
Number of sub-devices	Unsigned 8-bit integer	1
Total group identifiers	Unsigned 8-bit integer	1
Endpoint identifier	Unsigned 8-bit integer	0/1
Profile identifier	Unsigned 16-bit integer	0/2
Device identifier	Unsigned 16-bit integer	0/2
Version	Unsigned 8-bit integer	0/1
Group identifier count	Unsigned 8-bit integer	0/1

Figure 44 – Format of the scan response command frame

7.1.2.3.1.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN command frame received from the initiator.

7.1.2.3.1.2 RSSI correction field

The *RSSI correction* field is 8-bits in length and specifies a pre-programmed RSSI correction offset, specific to this device in the range 0x00 – 0x20.

7.1.2.3.1.3 ZigBee information field

The *ZigBee information* field is 8-bits in length and specifies information related to ZigBee. This field shall be formatted as illustrated in Figure 33.

7.1.2.3.1.4 ZLL information field

The *ZLL information* field is 8-bits in length and shall be formatted as illustrated in Figure 45.

Bits: 0	1	2-3	4	5	6-7
Factory new	Address assignment	Reserved	Touchlink initiator	Touchlink priority request	Reserved

Figure 45 – Format of the scan response ZLL information field

The *factory new* subfield is 1 bit in length and specifies whether the device is factory new. The value of this subfield shall be set to 1 to indicate the device is factory new or 0 otherwise.

The *address assignment* subfield is 1 bit in length and specifies whether the device is capable of assigning addresses. The value of this subfield shall be set to 1 to indicate the device is capable of assigning addresses or 0 otherwise.

The *touchlink initiator* subfield is 1 bit in length and specifies whether the device is initiating a touchlink operation. The value of this subfield shall be set to 1 to indicate the device is initiating a touchlink or 0 otherwise.

The *touchlink priority request* subfield is 1 bit in length and specifies that the target has requested some priority, possibly after a button push by the user. The value of this subfield shall be set to 1 to indicate that the device has requested priority or 0 otherwise.

7.1.2.3.1.5 Key bitmask field

The *key bitmask* field is 16-bits in length and specifies which keys (and hence which encryption algorithms) are supported by the device. The appropriate key shall be present on the device only if its corresponding bit is set to 1 otherwise the key is not supported. Bit *i* of the *key bitmask field* shall correspond to key index *i*, where $0 \le i \le 15$. (See also 8.7.)

7.1.2.3.1.6 Response identifier field

The *response identifier* field is 32-bits in length and specifies a random identifier for the response, used during the network key transfer mechanism.

7.1.2.3.1.7 Extended PAN identifier field

The *extended PAN identifier* field is 64-bits in length and specifies the extended PAN identifier of the device responding to the scan request.

If the *factory new* subfield of the *ZLL information* field indicates that the device is factory new and the value of this field is equal to zero, the target is not able to propose any network parameters. If the *factory new* subfield of the *ZLL information* field indicates that the device is factory new and the value of this field is not equal to zero, it can be used as the extended PAN identifier of a potential new network. Alternatively, if the *factory new* subfield of the *ZLL information* field indicates that the device is not factory new, this field indicates the current extended PAN identifier of the network on which the device operates.

7.1.2.3.1.8 Network update identifier field

The network update identifier field is 8-bits in length and specifies the current value of the *nwkUpdateId* attribute of the originator. If the factory new subfield of the ZLL information indicates the device to be in factory new mode, this field shall contain the value 0x00.

7.1.2.3.1.9 Logical channel field

The logical channel field is 8-bits in length and specifies the ZLL channel on which the device is operating.

If the *factory new* subfield of the *ZLL information* field indicates that the device is factory new and the value of the extended PAN identifier field is equal to zero, the target is not able to propose a logical channel for the network. If the *factory new* subfield of the *ZLL information* field indicates that the device is factory new and the value of the extended PAN identifier field is not equal to zero, this value can be used as the logical channel of a potential new network. Alternatively, if the *factory new* subfield of the *ZLL information* field indicates that the device is not factory new, this field indicates the current logical channel of the network on which the device operates.

7.1.2.3.1.10 PAN identifier field

The PAN identifier field is 16-bits in length and specifies the identifier of the PAN on which the device operates.

If the *factory new* subfield of the *ZLL information* field indicates that the device is factory new and the value of the extended PAN identifier field is equal to zero, the target is not able to propose a PAN identifier for the network. If the *factory new* subfield of the *ZLL information* field indicates that the device is factory new and the value of the extended PAN identifier field is not equal to zero, this value can be used as the PAN identifier of a potential new network. Alternatively, if the *factory new* subfield of the *ZLL information* field indicates that the device is not factory new, this field indicates the current PAN identifier of the network on which the device operates.

7.1.2.3.1.11 Network address field

The network address field is 16-bits in length and specifies the current network address of the device. If the factory new subfield of the ZLL information indicates the device to be in factory new mode, this value shall be set to 0xffff.

7.1.2.3.1.12 Number of sub-devices field

The number of sub-devices field is 8-bits in length and specifies the number of sub-devices (endpoints) supported by the device.

7.1.2.3.1.13 Total group identifiers field

The total group identifiers field is 8-bits in length and specifies the number of unique group identifiers that this device requires.

7.1.2.3.1.14 Endpoint identifier field

The endpoint identifier field is 8-bits in length and specifies the endpoint identifier of the sub-device. This field shall only be present when the number of sub-devices field is equal to 1.

7.1.2.3.1.15 Profile identifier field

The profile identifier field is 16-bits in length and specifies the profile identifier supported by the sub-device. This field shall only be present when the number of sub-devices field is equal to 1.

7.1.2.3.1.16 Device identifier field

The device identifier field is 16-bits in length and specifies the device identifier supported by the sub-device. This field shall only be present when the number of sub-devices field is equal to 1.

7.1.2.3.1.17 Version field

The version field is 8-bits in length and specifies the version of the device description supported by the sub-device on the endpoint specified by the *endpoint identifier* field. The least significant 4 bits of this value shall correspond to the *application device version* field of the appropriate simple descriptor (see also 8.1.3); the most significant 4 bits shall be set to 0x0.

This field shall only be present when the number of sub-devices field is equal to 1.

7.1.2.3.1.18 Group identifier count field

The group identifier count field is 8-bits in length and specifies the number of group identifiers required by the sub-device. This field shall only be present when the number of sub-devices field is equal to 1.

7.1.2.3.2 Device information response command frame

The *device information response* command frame is used to return information about the sub-devices supported by a node.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in the preceding *scan response* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 1 (server to client) and the command identifier shall be set to 0x03 (device information response).

The ZCL payload field shall contain the *device information response* command frame itself, formatted as illustrated in Figure 46.

Field name	**Data type**	**Octets**
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
Number of sub devices	Unsigned 8-bit integer	1
Start index	Unsigned 8-bit integer	1
Device information record count	Unsigned 8-bit integer	1
Device information record (see Figure 47)	Variable	(*n**16)

Figure 46 – Format of the device information response command frame

Field name	**Data type**	**Octets**
IEEE address	IEEE address	8
Endpoint identifier	Unsigned 8-bit integer	1
Profile identifier	Unsigned 16-bit integer	2
Device identifier	Unsigned 16-bit integer	2
Version	Unsigned 8-bit integer	1
Group identifier count	Unsigned 8-bit integer	1
Sort	Unsigned 8-bit integer	1

Figure 47 – Format of the device information record field

7.1.2.3.2.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN command frame received from the initiator.

7.1.2.3.2.2 Number of sub devices field

The *number of sub devices* field is 8-bits in length and specifies the number of sub devices contained in the device, as reported in the *scan response* inter-PAN command frame.

7.1.2.3.2.3 Start index field

The *start index* field is 8-bits in length and specifies the starting index into the device table from which to get device information. This value of this field shall be equal to the value of the start index field of the *device information request* command frame.

7.1.2.3.2.4 Device information record count field

The *device information record count* field is 8-bits in length and specifies the number *n* of device information records that follow. This value shall be in the range 0x00 – 0x05.

7.1.2.3.2.5 IEEE address field

The *IEEE address* field is 64-bits in length and shall contain the IEEE address of the device referred to by the device information record.

7.1.2.3.2.6 Endpoint identifier field

The *endpoint identifier* field is 8-bits in length and shall contain the endpoint identifier of the sub-device referred to by the device information record.

7.1.2.3.2.7 Profile identifier

The *profile identifier* field is 16-bits in length and shall contain the identifier of the profile supported by the sub-device referred to by the device information record.

7.1.2.3.2.8 Device identifier field

The *device identifier* field is 16-bits in length and shall contain the device identifier of the sub-device referred to by the device information record.

7.1.2.3.2.9 Version field

The *version* field is 8-bits in length and shall contain the version of the device description supported by the sub-device on the endpoint specified by the *endpoint identifier* field. The least significant 4 bits of this value shall correspond to the *application device version* field of the appropriate simple descriptor (see also 8.1.3); the most significant 4 bits shall be set to 0x0.

7.1.2.3.2.10 Group identifier count field

The *group identifier count* field is 8-bits in length and shall contain the number of unique group identifiers required by the sub-device referred to by the device information record.

7.1.2.3.2.11 Sort field

The *sort* field is 8-bits in length and shall contain the sorting order of the sub-device referred to by the device information record. This field is used to identify if a sorting of sub-devices is needed and what the order is, e.g., to sort the different lights in a luminaire in a selection list on the remote control. A value of zero shall indicate ‘not sorted’. Non-zero values shall indicate the order in the list, with the value 0x01 indicating the top of the list.

7.1.2.3.3 Network start response command frame

The *network start response* command frame is used by a router to respond to a *network start request* command frame received from an end device.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in the preceding *scan response* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 1 (server to client) and the command identifier shall be set to 0x11 (network start response).

The ZCL payload field shall contain the *network start response* command frame itself, formatted as illustrated in Figure 48.

Field name	Data type	Octets
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
Status	Unsigned 8-bit integer	1
Extended PAN identifier	IEEE address	8
Network update identifier	Unsigned 8-bit integer	1
Logical channel	Unsigned 8-bit integer	1
PAN identifier	Unsigned 16-bit integer	2

Figure 48 – Format of the network start response command frame

7.1.2.3.3.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN command frame received from the initiator.

7.1.2.3.3.2 Status field

The status field is 8-bits in length and shall contain the status code corresponding to the result of the network start request. This field shall be set to one of the values listed in Table 56.

Table 56 – Values of the status field of the network start response command frame

***Status* field value**	**Description**
0x00	Success
0x01	Failure
0x02 – 0xff	Reserved

7.1.2.3.3.3 Extended PAN identifier field

The *extended PAN identifier* field is 64-bits in length and shall contain the extended identifier of the new PAN.

7.1.2.3.3.4 Network update identifier field

The *network update identifier* field is 8-bits in length and shall be set to 0x00 in this version of the specification.

7.1.2.3.3.5 Logical channel field

The *logical channel* field is 8-bits in length and shall contain the ZLL channel used by the new network.

7.1.2.3.3.6 PAN identifier field

The *PAN identifier* field is 16-bits in length and shall contain the identifier of the new PAN.

7.1.2.3.4 Network join router response command frame

The *network join router response* command frame is used by a router to respond to a *network join router request* command frame received from a non factory new end device.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in the preceding *scan response* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 1 (server to client) and the command identifier shall be set to 0x13 (network join router response).

The ZCL payload field shall contain the *network join router response* command frame itself, formatted as illustrated in Figure 49.

Field name	Data type	Octets
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
Status	Unsigned 8-bit integer	1

Figure 49 – Format of the network join router response command frame

7.1.2.3.4.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN command frame received from the initiator.

7.1.2.3.4.2 Status field

The *status* field is 8-bits in length and shall contain the status code corresponding to the result of the network join router request. This field shall be set to one of the values listed in Table 57.

Table 57 – Values of the status field of the network join router response command frame

***Status* field value**	**Description**
0x00	Success
0x01	Failure
0x02 – 0xff	Reserved

7.1.2.3.5 Network join end device response command frame

The *network join end device response* command frame is used by a factory new end device to respond to a *network join end device request* command frame received from a non factory new end device.

This inter-PAN command shall be formatted according to the general inter-PAN frame format illustrated in Figure 56 with the following clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in the preceding *scan response* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 1 (server to client) and the command identifier shall be set to 0x15 (network join end device response).

The ZCL payload field shall contain the *network join end device response* command frame itself, formatted as illustrated in Figure 50.

Field name	**Data type**	**Octets**
Inter-PAN transaction identifier	Unsigned 32-bit integer	4
Status	Unsigned 8-bit integer	1

Figure 50 – Format of the network join end device response command frame

7.1.2.3.5.1 Transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN command frame received from the initiator.

7.1.2.3.5.2 Status field

The *status* field is 8-bits in length and shall contain the status code corresponding to the result of the network join end device request. This field shall be set to one of the values listed in Table 58.

Table 58 – Values of the status field of the network join end device response command frame

***Status* field value**	**Description**
0x00	Success
0x01	Failure
0x02 – 0xff	Reserved

7.1.2.3.6 Endpoint information command

The *endpoint information* command is used to inform the remote endpoint about the general information of the local endpoint. This command may be a trigger for the remote endpoint to get more information from the local device using the other commands described in this cluster.

Note: if the related endpoint(s) reside on sleeping end devices, the polling time and polling frequency must be chosen such that the exchange of information is done efficiently and in a timely manner.

The *endpoint information* command shall be sent using unicast transmission. On receipt of this command, the device shall respond using a ZCL default response (see [R2]) command.

This command shall be formatted as illustrated in Figure 51.

Octets	8	2	1	2	2	1
Data type	IEEE address	Unsigned 16-bit integer	Unsigned 8-bit integer	Unsigned 16-bit integer	Unsigned 16-bit integer	Unsigned 8-bit integer
Field name	IEEE address	Network address	Endpoint identifier	Profile identifier	Device identifier	Version

Figure 51 – Format of the endpoint information command

7.1.2.3.6.1 IEEE address field

The *IEEE address* field is 64-bits in length and specifies the IEEE address of the local device.

7.1.2.3.6.2 Network address field

The *network address* field is 16-bits in length and specifies the short network address of the local device.

7.1.2.3.6.3 Endpoint identifier field

The *endpoint identifier* field is 8-bits in length and specifies the identifier of the local endpoint.

7.1.2.3.6.4 Profile identifier field

The *profile identifier* field is 16-bits in length and specifies the identifier of the profile supported on the endpoint specified in the *endpoint identifier* field.

7.1.2.3.6.5 Device identifier field

The *device identifier* field is 16-bits in length and specifies the identifier of the device description supported on the endpoint specified in the *endpoint identifier* field.

7.1.2.3.6.6 Version field

The *version* field is 8-bits in length and specifies the version of the device description supported by the sub-device on the endpoint specified by the *endpoint identifier* field. The least significant 4 bits of this value shall correspond to the *application device version* field of the appropriate simple descriptor (see also 8.1.3); the most significant 4 bits shall be set to 0x0.

7.1.2.3.7 Get group identifiers response command

The *get group identifiers response* command allows a remote device to respond to the *get group identifiers request* command.

This command shall be formatted as illustrated in Figure 52.

Octets	1	1	1	(*n**3)
Data type	Unsigned 8-bit integer	Unsigned 8-bit integer	Unsigned 8-bit integer	Variable
Field name	Total	Start index	Count	Group information record list

Figure 52 – Format of the get group identifiers response command

7.1.2.3.7.1 Total field

The *total* field is 8-bits in length and specifies the total number of group identifiers supported by the device.

7.1.2.3.7.2 Start index field

The *start index* field is 8-bits in length and specifies the internal starting index from which the following group identifiers are taken and corresponds to the *start index* field of the *get group identifiers request* command.

7.1.2.3.7.3 Count field

The *count* field is 8-bits in length and specifies the number of entries in the *group information record list* field. If no entries are returned, this field shall be set to 0.

7.1.2.3.7.4 Group information record list field

The *group information record* field is (n * 24)-bits in length, where n is equal to the value of the *count* field, and specifies the requested group information. Each entry in this field shall be formatted as illustrated in Figure 53.

Octets	2	1
Data type	Unsigned 16-bit integer	Unsigned 8-bit integer
Field name	Group identifier	Group type

Figure 53 – Format of a group information record entry

The *group identifier* sub-field is 16-bits in length and specifies the identifier of the group described by this record.

The *group type* sub-field is 8-bits in length and has been introduced for future extensions. The group type shall indicate the meaning of a group in the user interface. In the current version of this specification, this value shall be set to 0x00.

7.1.2.3.8 Get endpoint list response command

The *get endpoint list response* command allows a remote device to respond to the *get endpoint list request* command.

This command shall be formatted as illustrated in Figure 54.

Octets	1	1	1	(n*8)
Data type	Unsigned 8-bit integer	Unsigned 8-bit integer	Unsigned 8-bit integer	Variable
Field name	Total	Start index	Count	Endpoint information record list (see Figure 55)

Figure 54 – Format of the get endpoint list response command

Octets	2	1	2	2	1
Data type	Unsigned 16-bit integer	Unsigned 8-bit integer	Unsigned 16-bit integer	Unsigned 16-bit integer	Unsigned 8-bit integer
Field name	Network address	Endpoint identifier	Profile identifier	Device identifier	Version

Figure 55 – Format of an endpoint information record entry

7.1.2.3.8.1 Total field

The *total* field is 8-bits in length and specifies the total number of endpoints supported by the device.

7.1.2.3.8.2 Start index field

The *start index* field is 8-bits in length and specifies the internal starting index from which the following list of endpoints are taken and corresponds to the *start index* field of the *get endpoint list request* command.

7.1.2.3.8.3 Count field

The *count* field is 8-bits in length and specifies the number of entries in the *endpoint information record list* field. If no entries are returned, this field shall be set to 0.

7.1.2.3.8.4 Network address field

The *network address* field is 16-bits in length and specifies the short network address of the device specified by the current endpoint information record.

7.1.2.3.8.5 Endpoint identifier field

The *endpoint identifier* field is 8-bits in length and specifies the identifier of the endpoint on the device specified by the *network address* field.

7.1.2.3.8.6 Profile identifier field

The *profile identifier* field is 16-bits in length and specifies the identifier of the profile supported on the endpoint, specified in the *endpoint identifier* field, on the device specified by the *network address* field.

7.1.2.3.8.7 Device identifier field

The *device identifier* field is 16-bits in length and specifies the identifier of the device description supported on the endpoint, specified in the *endpoint identifier* field, on the device specified by the *network address* field.

7.1.2.3.8.8 Version field

The *version* field is 8-bits in length and specifies the version of the device description supported by the sub-device on the endpoint, specified by the *endpoint identifier* field, on the device specified by the *network address* field. The least significant 4 bits of this value shall correspond to the *application device version* field of the appropriate simple descriptor (see also 8.1.3); the most significant 4 bits shall be set to 0x0.

7.1.3 Client

7.1.3.1 Attributes

The client has no attributes.

7.1.3.2 Commands received

The client receives the cluster specific response commands listed in Table 59. These commands are detailed in 7.1.2.3.

Table 59 – Commands received by the client side of the ZLL commissioning cluster

	Identifier	Description	Usage
Touchlink	0x01	Scan response	Mandatory
	0x03	Device information response	Mandatory
	0x11	Network start response	Mandatory
	0x13	Network join router response	Mandatory
	0x15	Network join end device response	Mandatory
Utility	0x40	Endpoint information	Optional
	0x41	Get group identifiers response	Mandatory if *get group identifiers request* command is generated. Optional otherwise.
	0x42	Get endpoint list response	Mandatory if *get endpoint list request* command is generated. Optional otherwise.

7.1.3.3 Commands generated

The client generates the cluster specific commands listed in Table 60. These commands are detailed in 7.1.2.2.

Table 60 – Commands generated by the client side of the ZLL commissioning cluster

	Identifier	Description	Usage
Touchlink	0x00	Scan request	Mandatory
	0x02	Device information request	Mandatory
	0x06	Identify request	Mandatory
	0x07	Reset to factory new request	Mandatory
	0x10	Network start request	Mandatory
	0x12	Network join router request	Mandatory
	0x14	Network join end device request	Mandatory
	0x16	Network update request	Mandatory
Utility	0x41	Get group identifiers request	Optional
	0x42	Get endpoint list request	Optional

8 Functional description

8.1 General

8.1.1 ZigBee stack profile

The ZigBee Light Link profile shall use the ZigBee-Pro stack profile.

8.1.2 Channels

A ZLL device shall be able to operate on all channels available at 2.4GHz, numbered from 11 to 26. When operating on channel 26, the transmission power may be reduced in order to comply with FCC regulations.

Within this range, two sets of channels shall be defined. The ***primary*** ZLL channel set shall consist of channels 11, 15, 20 and 25 and shall be used in preference for commissioning and normal operations. The ***secondary*** ZLL channel set shall consist of channels 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24 and 26 and can be used as a backup to allow the ZLL device to connect to a non-ZLL network.

8.1.3 Application device version

When a device conforms to this version of the specification, the *application device version* field of all simple descriptors shall be set to 0x2.

On receipt of command frames which contain a version field, i.e. scan response, device information response, endpoint information and get endpoint list response, the recipient shall handle the device according to its indicated version, i.e., if the indicated version is greater than or equal to that of the recipient, it shall assume it is fully compatible and if the indicated version is less than that of the recipient, it shall handle the device according to the appropriate version of the profile specification.

8.1.4 Profile identifier

The profile identifier for this profile shall be 0xc05e.

In order to provide a foundation for future profile interoperability with devices supporting the home automation profile, ZCL commands addressing clusters other than the *ZLL commissioning* cluster or those commands in the commissioning utility command set of the *ZLL commissioning* cluster, shall use the profile identifier 0x0104.

8.1.5 ZDO requirements

This profile requires the mandatory ZDO feature set of a compliant ZigBee-pro stack.

8.1.6 Startup attribute set

In order to ensure interoperability with other ZigBee devices, all ZLL devices should implement the compatible startup attribute set (SAS) specified in this sub-clause. ZLL devices can join other non-ZLL ZigBee networks and allow non-ZLL devices to join ZLL networks under application control.

This sub-clause separates out startup and security attributes. All other attributes are set by the stack profile.

8.1.6.1 Startup attributes

The SAS for general startup is specified in Table 61.

Table 61 – SAS for startup attributes

Attributes	**Value**	**Notes**
Short address	0xffff	No short address.
Extended PAN identifier	0x00[x8]	No extended PAN identifier.
PAN identifier	0xffff	Device has not yet joined a network.
Channel mask	All channels in the frequency band with a commissioning preference to the ZLL channel set.	If needed, the power transmitted by the device on channel 26 can be lowered to comply with FCC regulations.
Protocol version	0x02	ZigBee specification 2007.
Stack profile	0x02	ZigBee-Pro.
Startup control	0x03	Be able to join a network by association, under application control.
Permit join	Disabled	Enabled under application control.

8.1.6.2 Security attributes

The SAS for security is specified in Table 62.

Table 62 – SAS for security attributes

Attributes	**Value**	**Notes**
Trust centre address	0x00[x8]	Trust centre address is unknown.
Master key	0x00[x16]	Master key is unspecified.
Network key type	0x01	Standard security.
Network key sequence number	0x00	-
Network key	0x00[x16]	Network key is unspecified.
Preconfigured link key	0x00[x16]	Preconfigured link key is unspecified.
Trust centre link key	0x5a 0x69 0x67 0x42 0x65 0x65 0x41 0x6c 0x6c 0x69 0x61 0x6e 0x63 0x65 0x30 0x39	Default key for communicating with a trust centre.
Use insecure join	True	Use insecure join as a fallback option.

Attributes	Value	Notes
Trust centre network key	Trust centre will pick the key.	

8.1.7 Device Information Table

Each ZLL device shall contain a *device information table* that holds the necessary (static) application information that is exchanged during device discovery (e.g. touch-linking). Each entry gives information about a so called sub-device which is a self contained device such as a dimmable light. In ZigBee terms, a sub-device resides as a device application on an endpoint. Each entry shall be formatted as illustrated in Table 63.

Table 63 – Format of the device information table

Field name	Data type	Bits
IEEE address	IEEE address	64
Endpoint identifier	Unsigned 8-bit integer	8
Profile identifier	Unsigned 16-bit integer	16
Device identifier	Unsigned 16-bit integer	16
Device version	Unsigned 4-bit integer	4
Reserved	-	4
Number of groups identifiers	Unsigned 8-bit integer	8
Sort tag	Unsigned 8-bit integer	8

8.1.7.1 IEEE address field

The *IEEE address* field is 64-bits in length and specifies the unique IEEE identifier for each single node.

8.1.7.2 Endpoint identifier field

The *endpoint identifier* field is 8-bits in length and specifies the identifier of the endpoint on which the sub-device is implemented. This value is determined by the application and can be freely chosen by the application in the range 0x01 – 0xf0. The ZLL does not rely on any fixed endpoint identifier for its applications.

8.1.7.3 Profile identifier field

The *profile identifier* field is 16-bits in length and specifies the identifier of the profile supported by the sub-device. This value shall correspond to the *application profile identifier* field of the simple descriptor.

8.1.7.4 Device identifier field

The *device identifier* field is 16-bits in length and specifies the identifier of the device description supported by the sub-device. This value shall correspond to the *application device identifier* field of the simple descriptor.

8.1.7.5 Device version field

The *device version* field is 4-bits in length and specifies the version of the device description supported by the sub-device. This value shall correspond to the *application device version* field of the simple descriptor (see also 8.1.3).

8.1.7.6 Number of group identifiers field

The *number of group identifiers* field is 8-bits in length and specifies the number of unique group identifiers required by the application on that specific endpoint.

8.1.7.7 Sort tag field

The *sort tag* field is 8-bits in length and specifies a sorting of the sub-devices, if required. A value of 0x00 indicates that the field is not sorted. Other values indicate the order in the list.

8.1.8 Constants

The constants that define the characteristics of the ZLL profile are listed in Table 64.

Table 64 – ZLL profile constants

Constant	Description	Value
aplcInterPANTransIdLifetime	The maximum length of time an inter-PAN transaction identifier remains valid.	8s
aplcMaxPermitJoinDuration	The maximum number of seconds a factory new router will enable its permit join flag after its initial network scans did not find any suitable networks.	60
aplcMaxLostParentRetryAttempts	The maximum number of attempts an end device shall use to reacquire a lost parent before going into a dormant state. The end device can be reactivated under application control.	10
aplcMaxPollInterval	The maximum time an end device is permitted to use as its poll interval without first attempting a network rejoin before transmitting any application data.	1 hour
aplcMinChildPersistenceTime	The minimum time a child must persist in the child table of its parent before it can be aged out.	(4 * *aplcMax-PollInterval)*
aplcMinStartupDelayTime	The length of time an initiator waits to ensure that the recipient has completed its startup procedure.	2s
aplcRxWindowDuration	The maximum duration that a device leaves its receiver enabled during the joining procedure for subsequent configuration information.	5s
aplcScanTimeBaseDuration	The base duration for a scan operation during which the receiver is enabled for scan responses.	0.25s

8.1.9 ZLL profile attributes

The ZLL profile defines internal attributes required to allow a device to manage the way the ZLL profile operates. These attributes are summarized in Table 65.

Table 65 – ZLL profile attributes

Attribute	Type	Ref	Default
aplFreeNwkAddrRangeBegin	Unsigned 16-bit integer	8.1.9.1	0x0001
aplFreeNwkAddrRangeEnd	Unsigned 16-bit integer	8.1.9.2	0xfff7
aplFreeGroupIDRangeBegin	Unsigned 16-bit integer	8.1.9.3	0x0001
aplFreeGroupIDRangeEnd	Unsigned 16-bit integer	8.1.9.4	0xfeff

8.1.9.1 *aplFreeNwkAddrRangeBegin* attribute

The *aplFreeNwkAddrRangeBegin* attribute is an unsigned 16-bit integer in the range 0x0000 – 0xfff7 and contains the starting value of the free network address range for address assignment capable devices. Address assignment capable devices should use and maintain this value when assigning addresses via touchlink commissioning. If the device is not address assignment capable or it has joined a non-ZLL network through classical ZigBee joining mechanisms, this attribute should be set to 0x0000.

8.1.9.2 *aplFreeNwkAddrRangeEnd* attribute

The *aplFreeNwkAddrRangeEnd* attribute is an unsigned 16-bit integer in the range 0x0000 – 0xfff7 and contains the end value of the free network address range for address assignment capable devices. Address assignment capable devices should use and maintain this value when assigning addresses via touchlink commissioning. If the device is not address assignment capable or it has joined a non-ZLL network through classical ZigBee joining mechanisms, this attribute should be set to 0x0000.

8.1.9.3 *aplFreeGroupIDRangeBegin* attribute

The *aplFreeGroupIDRangeBegin* attribute is an unsigned 16-bit integer in the range 0x0000 – 0xfeff and contains the starting value of the free group identifier range for address assignment capable devices. Address assignment capable devices should use and maintain this value when assigning group identifiers via touchlink commissioning. If the device is not address assignment capable or it has joined a non-ZLL network through classical ZigBee joining mechanisms, this attribute should be set to 0x0000.

8.1.9.4 *aplFreeGroupIDRangeEnd* attribute

The *aplFreeGroupIDRangeEnd* attribute is an unsigned 16-bit integer in the range 0x0000 – 0xfeff and contains the end value of the free group identifier range for address assignment capable devices. Address assignment capable devices should use and maintain this value when assigning group identifiers via touchlink commissioning. If the device is not address assignment capable or it has joined a non-ZLL network through classical ZigBee joining mechanisms, this attribute should be set to 0x0000.

8.1.10 Inter-PAN frame format

The inter-PAN mechanism used by the ZLL profile was first used in the ZigBee Smart Energy profile specification (see [R3]). When using the inter-PAN frame format for ZLL commissioning, frames shall be either broadcast or unicast directly to the recipient, depending on the frame (i.e. indirect transmissions are not permitted). The general format of an inter-PAN frame is illustrated in Figure 56.

Group	Field name	Octets	Description
MAC header	Frame control	2	Frame Type = 0b001 Security Enabled = 0 Frame Pending = 0 ACK Request = 0 (no ACK requested) or 1 (ACK requested) Intra-PAN = 0 Dest. Addressing Mode = 0b10 (short address) or 0b11 (extended address) Frame Version = As appropriate Source Addressing Mode = 0b11
	Sequence number	1	As appropriate
	Destination PAN ID	2	0xffff
	Destination address	2/8	0xffff if broadcast IEEE address of destination otherwise
	Source PAN ID	2	As appropriate
	Source address	8	IEEE address of source
NWK header	Frame control	2	Frame type = 0b11 Protocol version = as appropriate Remaining sub-fields ≡ 0
APS header	Frame control	1	Frame type = 0b11 Delivery mode = 0b00 (unicast) or 0b10 (broadcast) ACK format = 0 Security = 0 ACK request = 0 Extended header present = 0
	Group address	0	Not included
	Cluster identifier	2	0x1000
	Profile identifier	2	0xc05e
ZCL header	Frame control	1	Frame type = 0b01 Manufacturer specific = As appropriate[8] Direction = 0 (client to server) or 1 (server to client) Disable default response = 1
	Manufacturer code	0/2	Included only if the manufacturer specific sub-field is set to 1.
	Transaction sequence number	1	Incremented for every transmission of a command
	Command identifier	1	See clause 7.1.
ZCL payload	Command payload	Variable	See clause 7.1.
MAC footer	Frame check sequence	2	As appropriate for the frame

Figure 56 – General format of an inter-PAN frame

8.1.11 Inter-PAN transaction identifier

All ZLL inter-PAN command frames shall carry a 32-bit transaction identifier.

The transaction identifier shall be created by the initiator of a *scan request* inter-PAN command frame and shall be random, non zero and non sequential. Related inter-PAN command frames which follow

[8] When using the inter-PAN command frames defined in the ZLL commissioning cluster (see 7.1), the manufacturer specific sub-field of the ZLL header frame control field shall be set to 0.

the *scan request*, i.e., *scan response*, *device information request/response*, *identify request*, *reset to factory new request*, *network start request/response*, *network join router request/response* and *network join end device request/response* define the scope of a transaction (illustrated in Figure 57) and shall carry the same transaction identifier as was defined in the *scan request*. While within the scope of a transaction (and for at most *aplcInterPANTransIdLifetime*), the transaction identifier is said to be valid.

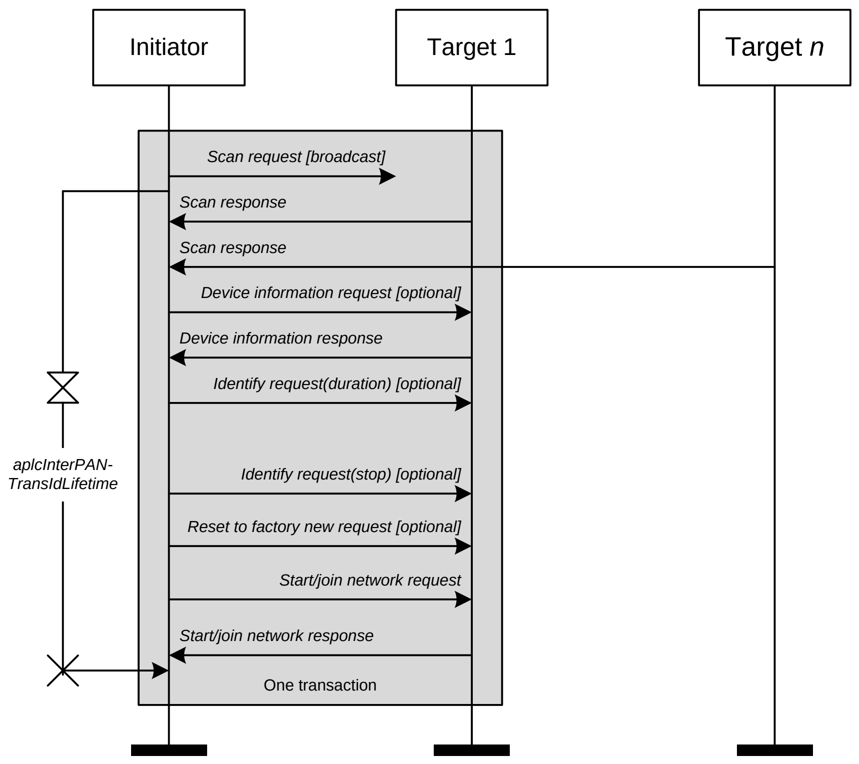

Initiator
Target 1
Target *n*
Scan request [broadcast]
Scan response
Scan response
Device information request [optional]
Device information response
Identify request(duration) [optional]
aplcInterPAN-TransIdLifetime
Identify request(stop) [optional]
Reset to factory new request [optional]
Start/join network request
Start/join network response
One transaction

Figure 57 – Scope of an inter-PAN transaction

If a target, receiving a *scan request* inter-PAN command frame, is a sleeping end device, it shall enable its receiver while the transaction identifier is valid or for at most *aplcInterPANTransIdLifetime* seconds after reception of the original *scan request* inter-PAN command frame. A device may disable its receiver before *aplcInterPANTransIdLifetime* seconds have elapsed if the transaction has successfully completed and the device has started or joined the network.

During a transaction, a device shall only accept inter-PAN command frames that contain a valid transaction identifier, i.e., inter-PAN command frames from within a transaction that have the same transaction identifier as was received in the *scan request* inter-PAN command frame, unless a device wants to start a new transaction after receiving a new *scan request* inter-PAN command frame from the same or another initiator carrying a new transaction identifier.

8.1.12 Commissioning scenarios

Commissioning between ZLL devices is performed from an *initiator* to a *target*, both of which can be implemented from either an end device or a router. The commissioning mechanisms depend on whether the initiator is factory new or non factory new. If the initiator is factory new, it requests a new network to be started and if the initiator is non factory new it requests the target to join its network. If the target is non factory new and already part of a network, it can be *stolen* onto the network of the initiator. However, the target can decide whether to accept a request to start a new network or join an existing network when requested to do so by the initiator. If the initiator is a factory new end device, it must be commissioned with a router target so that a new network can be formed.

8.2 ZigBee-pro stack requirements

8.2.1 Initialization NIB settings

A device operating according to the ZLL profile shall configure the NIB attributes listed in Table 66. All other NIB attributes shall be set to their default values.

Table 66 – ZLL profile initial NIB attribute settings

NIB attribute	Identifier	Initial value
nwkUseMulticast	0x9b	FALSE

8.2.2 End-device rejoining

If an end-device has lost communication with the router which is acting as its parent, it shall try to reacquire its parent by polling for at most *aplcMaxLostParentRetryAttempts* times or until the poll was successful.

If its parent is reacquired within *aplcMaxLostParentRetryAttempts* poll attempts, the end-device returns to its normal operating state.

If its parent is not reacquired after *aplcMaxLostParentRetryAttempts* poll attempts, the end device shall attempt to find a new parent on the same network. To do this, the end device shall perform a primary network discovery by issuing the NLME-NETWORK-DISCOVERY.request primitive to the NWK layer over the primary channel set (see 8.1.2) and the *ScanDuration* parameter set to 4 (240 milliseconds scan per channel).

On receipt of the NLME-NETWORK-DISCOVERY.confirm primitive, the end device shall determine if another router exists on the same network. If a router is found on the same network, the end device shall perform a secure NWK rejoin and then transmit a Device_annce command frame, as defined in section 2.4.3.1.11 of [R1] . This command frame shall be broadcast to all devices for which *macRxOnWhenIdle* is equal to True (i.e. a network address of 0xfffd). The device shall ensure that the allocate address sub field of the capability field of the Device_annce command frame is set to 0, indicating that a new address allocation is not required.

If a router is not found on the same network and *apsTrustCenterAddress* is not equal to 0xffffffffffffffff, the end device shall perform a secondary network discovery by issuing the NLME-NETWORK-DISCOVERY.request primitive to the NWK layer over the secondary channel set (see 8.1.2) and the *ScanDuration* parameter set to 4 (240 milliseconds scan per channel).

On receipt of the NLME-NETWORK-DISCOVERY.confirm primitive, the end device shall determine if another router exists on the same network. If a router is found on the same network, the end device shall perform a secure NWK rejoin and then transmit a Device_annce command frame, as defined in section 2.4.3.1.11 of [R1] . This command frame shall be broadcast to all devices for which *macRxOnWhenIdle* is equal to True (i.e. a network address of 0xfffd). The device shall ensure that the allocate address sub field of the capability field of the Device_annce command frame is set to 0, indicating that a new address allocation is not required.

If the end device could not find another router on the same network, it shall enter a low power, dormant state. The end device can be reactivated under application control at which point it shall repeat its rejoining attempt.

8.2.3 Link status messages

A non-factory new device acting as a ZigBee router shall set *nwkLinkStatusPeriod* to 0x0f; this results in a link status message being sent every 15 seconds.

8.2.4 ZigBee device announcement

In order to avoid that newly joined devices send empty ZigBee link status command frames, resulting in incorrect routing information and effectively leading to not being able to communicate to this newly joined device within seconds after joining, router devices shall observe special handling of ZigBee Device_annce command frames.

On receipt of a ZigBee Device_annce command frame, initiated directly from another ZigBee router (i.e. the MAC source address of the Device_annce command frame is the same as the device indicated in the Device_annce command frame itself), a router device shall check whether the device sending the ZigBee Device_annce command frame is present in its neighbor table. If the neighbor table is not full, an entry does not exist in the neighbor table and the router device has at least one end device as a child, it shall immediately send a ZigBee link status command frame, containing a link status entry corresponding to the device sending the ZigBee Device_annce command frame.

8.2.5 End device polling

An end device may choose to periodically poll its parent at an implementation specified interval. However, if the device polls at a rate greater than *aplcMaxPollInterval* or does not poll at all, the device shall ensure that it connected to a network before transmitting any application commands. To do this, the end device shall transmit a NWK rejoin command frame to its assumed parent.

If the NWK rejoin command frame is successfully acknowledged by the assumed parent using a NWK rejoin response command frame, the end device shall consider itself part of a network and not issue a device_annce command frame.

If the NWK rejoin command frame is not successful, the end device shall continue its network rejoin attempt via a channel scan, as described in [R1] . If the end device successfully rejoins the network it shall transmit a device_annce command frame.

Note that if the end device has been in contact with its parent within a time less than or equal to *aplcMaxPollInterval*, it does not need to attempt a rejoin before it transmits an application command.

8.2.6 Child table maintenance

The contents of the child table shall be preserved through a power cycle.

If a parent device does not receive a command frame (e.g. a data request command frame) from one of its children for longer than *aplcMinChildPersistenceTime*, it may choose to remove that device from its child table. The algorithm to determine when to age out a child device is out of scope of this specification.

Since it is possible in a consumer environment for a parent device to be powered off by the user, the network will automatically recover and any children of that parent will seek new parents. In the event that the original parent is powered back on, it may see traffic from what it believes is one of its children. As that child will have found a new parent, a conflict could arise in the network.

On receipt of a unicast message from a device which is listed in its child table, the parent device shall verify that this device is indeed still one of its children. If the MAC level source address in the message is equal to the NWK level source address, then the device shall be assumed to be a valid child and its message processed accordingly. If the MAC level source address is not equal to the NWK level source address, then the device shall be assumed to be a child of another device. In this case, the device shall be removed from the child table. The frame shall then be further processed in the normal way.

On receipt of a non unicast (i.e. broadcast or multicast) message, the router device shall process the frame in the normal way for non unicast frames.

On receipt of a data request command frame from a device that is no longer in its child table, a parent device may send a NWK leave command frame back to the device. Furthermore, if the IEEE address of the device is not known, it shall be permitted to send the NWK leave command frame using the short network address as used by the device.

8.3 Device startup

8.3.1 End-device

At startup, a factory new end device shall not instigate the ZigBee join procedure to look for a network. Instead, the end device shall select a primary ZLL channel and wait for an application request to begin a touchlink procedure. Note that the application should decide the best way to instigate the touchlink procedure, e.g. it may start an initiator discovery, enable the receiver or provide a combination of the two.

If the end device is not factory new, it shall resume ZigBee functionality based on the information stored in NVRAM by performing the ZigBee network rejoin procedure. After the end device has successfully joined, it shall transmit a Device_annce command frame, as defined in section 2.4.3.1.11 of [R1] . This command frame shall be broadcast to all devices for which *macRxOnWhenIdle* is equal to True (i.e. a network address of 0xfffd).

8.3.2 Router

If the router is factory new, it shall first ensure that *macRxOnWhenIdle* is set to True and then randomly select one of the primary ZLL channels, tune to it and then enable its receiver, waiting for a touchlink command.

If the router is not factory new, it shall resume ZigBee functionality based on the information stored in NVRAM by performing the ZigBee start router procedure. It does this by issuing the NLME-START-ROUTER.request primitive to the NWK layer. After the router has successfully started, it shall transmit a *Device_annce* command frame, as defined in section 2.4.3.1.11 of [R1] . This command frame shall be broadcast to all devices for which *macRxOnWhenIdle* is equal to True (i.e. a network address of 0xfffd).

8.4 Touchlink commissioning

8.4.1 Device discovery

8.4.1.1 Initiator procedure

Device discovery shall only be initiated by address assignment capable devices. To perform device discovery, the initiator shall broadcast eight inter-PAN *scan request* command frames (see 7.1.2.2.1), spaced at an interval of *aplcScanTimeBaseDuration* seconds, with the *touchlink initiator* sub-field of the ZLL information field set to 1 and with a nominal output power of 0dBm.

The eight inter-PAN *scan request* command frames shall be transmitted as follows. The initiator shall switch to the first primary ZLL channel (i.e., channel 11) and broadcast five consecutive *scan request* inter-PAN command frames. The initiator shall then switch to each of the remaining primary ZLL channels in turn (i.e., channels 15, 20 and 25, respectively) and broadcast a single *scan request* inter-PAN command frame on each channel. After each transmission, the initiator shall wait *aplcScanTimeBaseDuration* seconds to receive any responses.

If an extended scan is required (i.e. for a reset to factory new or if the initiator is part of a non-ZLL network), the initiator shall switch to each of the secondary ZLL channels in turn and broadcast a single *scan request* inter-PAN command frame on each channel. After each transmission, the initiator shall wait *aplcScanTimeBaseDuration* seconds to receive any responses.

If any responses are received with a valid transaction identifier, the initiator may request information about the sub-devices of a target by unicasting a *device information request* inter-PAN command frame (see 7.1.2.2.2). However, this is not necessary if a target has only one sub-device since its information is entirely contained in the *scan response* inter-PAN command frame. If any responses are received with a valid transaction identifier and the touchlink priority request bit of the ZLL information field of the *scan response* inter-PAN command frame is set to 1, the initiator may consider giving priority processing to those devices.

The device discovery operation may be aborted at any time. Since no device parameters such as network settings are altered, this step is non-intrusive for the devices involved. A new device discovery operation may be started at any time via a new *scan request* inter-PAN command frame with a different transaction identifier.

If, during its scan, a non factory new initiator receives another *scan request* inter-PAN command frame from a factory new target, it shall be ignored.

8.4.1.2 Target procedure

If the target is an end device, it may first need to be woken so that it can enable its receiver and respond to the scan from the initiator. This should be accomplished by application specific means, e.g. through some button press which enables the receiver or by initiating a scan itself. If a scan is initiated by a factory new target, it shall defer to a *scan request* inter-PAN command frame from a non factory new initiator and respond accordingly.

Devices receiving the *scan request* inter-PAN command frame may choose whether to respond and, if so, the device shall unicast a *scan response* inter-PAN command frame (see 7.1.2.3.1) back to the initiator on the same channel on which the *scan request* was received. If the *scan response* inter-PAN command frame was successfully received by the initiator, the target shall ignore further *scan request* inter-PAN command frames from the initiator which have the same transaction identifier.

A device shall only respond to a received *scan request* inter-PAN command frame if its RSSI is above a certain manufacturer specific threshold and the link initiator sub-field of the ZLL information field is set to 1. The *scan response* inter-PAN command frame shall then be transmitted with a nominal output power of 0dBm.

The target shall set the RSSI correction field of the *scan response* inter-PAN command frame to a certain product specific RSSI correction value in order to compensate for RF signals losses between the radio and the outer side of a product; the initiator can then use this value from each discovered target to select an appropriate device.

If the target device wishes to be considered as a priority by the initiator during touchlinking (e.g. if the target is power constrained and is responding to the scan following a button press from the user), the touchlink priority request bit of the ZLL information field may be to 1.

The target shall set the response identifier field to a random (non-sequential) value. If not factory new, the target shall set the extended PAN identifier, network update identifier, logical channel, PAN identifier and network address fields to the corresponding values determined by the router during its startup phase from NVRAM. All future inter-PAN communication between the initiator and the target shall be conducted on the channel indicated in the logical channel field.

On receipt of a *device information request* inter-PAN command frame, a target shall respond to it by unicasting a corresponding *device information response* inter-PAN command frame (see 7.1.2.3.2).

8.4.2 Identify

The device discovery operation can result in a list of potential devices from which the application can select one or more devices for further processing. In order to support a user confirmation, the initiator has the possibility to have the selected devices identify themselves.

8.4.2.1 Initiator procedure

To request a device identify itself, an initiator shall first follow the procedure for device discovery as specified in 8.4.1.1. Once within a transaction, the initiator shall generate and transmit an *identify request* inter-PAN command frame (see 7.1.2.2.3) to the appropriate discovered target device.

Note that the *identify request* inter-PAN command frame shall contain the same *inter-PAN transaction identifier* that was used in the *scan request* inter-PAN command frame during device discovery. If the initiator wishes to send a further *identify request* inter-PAN command frame, for example, to stop the identify operation, it must do so within the active transaction time. If this is not possible, a new device discovery operation must be performed.

8.4.2.2 Target procedure

On receipt of an *identify request* inter-PAN command frame with a valid transaction identifier (i.e. immediately following a device discovery), the target should identify itself in an application specific way (e.g., by flashing a lamp) for a time depending on the value of the *identify time* field. If the value of this field is 0x0000, the device shall immediately stop its identify operation. If the value of this field is 0xffff, the device shall identify itself for an application specific time. For all other values, the device shall identify itself for a time equal to this value in units of 1 second.

If an *identify request* inter-PAN command frame is received with an invalid transaction identifier (i.e. the frame was not received within the current active transaction), it shall discard the frame and perform no further processing.

The target shall generate no response to an *identify request* inter-PAN command frame.

Note that the identify operation shall not block the target from receiving further commands, such as a request to stop identifying.

8.4.3 Starting a new network

This section describes how a network can be formed between a factory-new end device initiator and a router target.

8.4.3.1 Initiator procedure

During device discovery, the initiator will have found an appropriate target. To start a new network, the initiator shall generate a *network start request* inter-PAN command frame (see 7.1.2.2.5) as follows.

If the initiator wishes to specify the PAN identifier, extended PAN identifier and logical channel for the new network it shall specify these values in the corresponding fields of the *network start request* inter-PAN command frame. Otherwise, the initiator shall set these fields to zero, indicating that they must be determined by the target.

The initiator shall set the key index and encrypted network key fields of the *network start request* inter-PAN command frame accordingly to describe the ZigBee network key to be used for securing the network.

The initiator shall set the network address field of the *network start request* inter-PAN command frame to the selected network address with which the target shall operate on the network. If the value of the *aplFreeNwkAddrRangeBegin* attribute is equal to 0x0000, the initiator shall stochastically generate an address according to the classical ZigBee mechanism. If the value of the *aplFreeNwkAddrRangeBegin* attribute is not equal to 0x0000, the initiator shall give the target the address *aplFreeNwkAddrRange-Begin* and increment the value. The target shall not change the network address unless it leaves the network and joins another or if required to do so in order to resolve an address conflict.

If the target requested a set of group identifiers in its *scan response* inter-PAN command frame and the value of the *aplFreeGroupIDRangeBegin* attribute is not equal to 0x0000, the initiator shall allocate a range of group identifiers for the target and set the group identifiers begin and group identifiers end fields of the *network start request* inter-PAN command frame accordingly. If the target requested a set of group identifiers in its *scan response* inter-PAN command frame and the value of the *aplFreeGroupIDRangeBegin* attribute is equal to 0x0000, the initiator shall set the group identifiers begin and group identifiers end fields of the *network start request* inter-PAN command frame to 0x0000.

If the target indicated that it was address assignment capable in its *scan response* inter-PAN command frame and the value of the *aplFreeNwkAddrRangeBegin* attribute is not equal to 0x0000, the initiator shall allocate a range of network addresses and group identifiers that the target can use for its own purposes and set the free network address range begin, free network address range end, free group identifier range begin and free group identifier range end fields of the *network start request* inter-PAN command frame accordingly. If the target indicated that it was address assignment capable in its *scan response* inter-PAN command frame and the value of the *aplFreeNwkAddrRangeBegin* attribute is equal to 0x0000, the initiator shall set the free network address range begin, free network address range end, free group identifier range begin and free group identifier range end fields of the *network start request* inter-PAN command frame to 0x0000.

Finally, the initiator shall set the initiator IEEE address and initiator network address fields of the *network start request* inter-PAN command frame to its IEEE address and the network address it will use on the new network, respectively.

Once the *network start request* inter-PAN command frame has been generated, the initiator shall unicast it to the selected target. It shall then enable its receiver and wait for at *aplcRxWindowDuration* seconds or until a *network start response* inter-PAN command frame is received with a valid transaction identifier. If a *network start response* inter-PAN command frame is not received within *aplcRxWindowDuration* seconds or if a *network start response* inter-PAN command frame is received within *aplcRxWindowDuration* seconds but with a non-zero value in the *Status* parameter, the initiator shall terminate the operation with this target. If there are no further targets to select, the initiator shall terminate the operation and perform no further processing.

On receipt of a *network start response* inter-PAN command frame with a valid transaction identifier from the intended target and if the network parameters were to be determined by the target, the initiator shall copy these parameters to its network information base. The initiator shall then wait at least *aplcMinStartupDelayTime* seconds to allow the target to start the network correctly.

The initiator shall then perform a network rejoin request by issuing the NLME-JOIN.request primitive to the NWK layer, ensuring the *RejoinNetwork* parameter is set to indicate that the device is joining the network using the NWK rejoining procedure. If the network rejoin was successful (indicated by the reception of the NLME-JOIN.confirm), the initiator shall transmit a Device_annce command frame, as defined in section 2.4.3.1.11 of [R1] . This command frame shall be broadcast to all devices for which *macRxOnWhenIdle* is equal to True (i.e. a network address of 0xfffd). The initiator and target can then use the network to communicate.

8.4.3.2 Target procedure

On receipt of a *network start request* inter-PAN command frame with a valid transaction identifier, the target shall decide by application specific means whether to allow itself to start a new network. If the target decides not to start a new network, it shall generate a *network start response* inter-PAN command frame (see 7.1.2.3.4) with a status indicating failure and transmit it back to the initiator using the unicast data service; the target shall then perform no further processing.

If the target decides to start a new network, it shall check the PAN identifier, extended PAN identifier and logical channel fields. For each of these fields, if the value of the field is equal to zero, the target shall determine an appropriate value. In order to verify that the PAN identifier and extended PAN identifier are unique, the target shall issue the NLME-NETWORK_DISCOVERY.request primitive to the NWK layer, over the primary ZLL channels, and wait for the corresponding NLME-NETWORK-DISCOVERY.confirm primitive, evaluating the results. The target shall then set *apsTrustCenterAddress* to 0xffffffffffffffff.

The target shall then generate a *network start response* inter-PAN command frame with a status indicating success and transmit it back to the initiator using the unicast data service.

If the target is not factory new, it shall perform a leave request on its old network by issuing the NLME-LEAVE.request primitive to the NWK layer and wait for the corresponding NLME-LEAVE.confirm response. The target shall ensure that its old network parameters are reset to their default values, as required by the leave procedure. The target shall then copy the new network parameters to its network information base and start operating on the new network by issuing the NLME-START-ROUTER.request primitive to the NWK layer.

After the router has successfully started, it shall transmit a Device_annce command frame, as defined in section 2.4.3.1.11 of [R1] . This command frame shall be broadcast to all devices for which *macRxOnWhenIdle* is equal to True (i.e. a network address of 0xfffd).

In order to allow direct communication via the ZigBee network between the initiator and the target, the target shall finally perform a ZigBee direct join procedure by issuing the NLME-DIRECT-JOIN.request primitive to the NWK layer in order to create an entry in the neighbor table with the IEEE address and the network address of the initiator.

8.4.3.3 Sequence chart for starting a new network

The sequence of events for starting a new network (described in the previous sub-clauses) is illustrated in Figure 58.

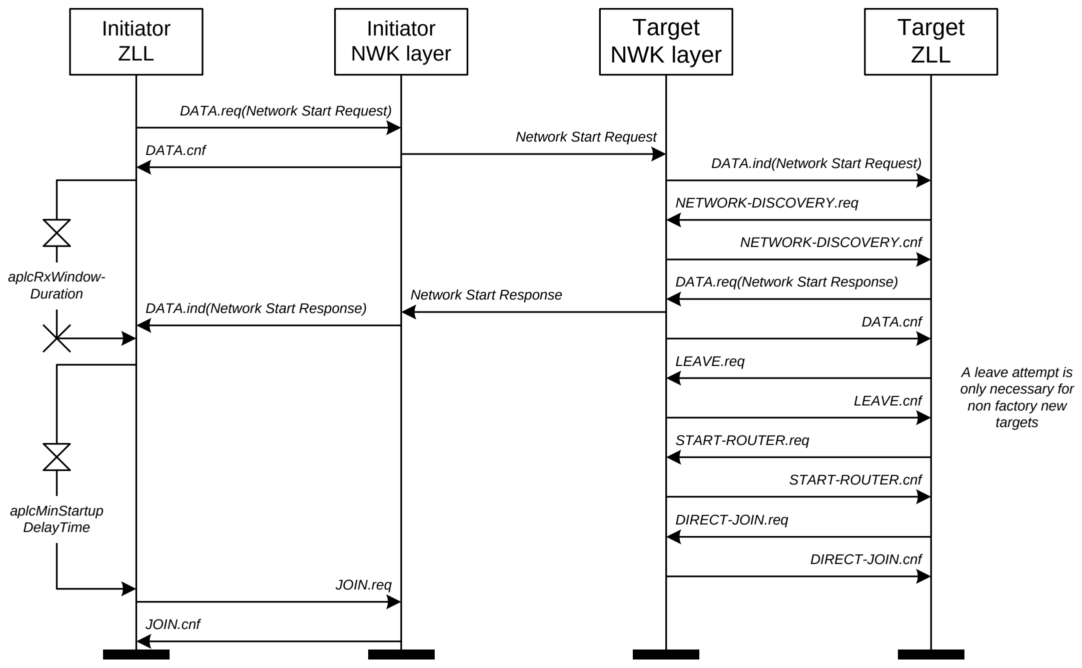


Figure 58 – Starting a new network

8.4.4 Joining routers to the network

A router target shall be joined to the network by an initiator that is already part of a network.

8.4.4.1 Initiator procedure

During a device discovery operation, the initiator will have found the router target. To add the target to the network, the initiator shall generate a *network join router request* inter-PAN command frame (see 7.1.2.2.6) as follows.

The initiator shall set the extended PAN identifier, network update identifier, logical channel and PAN identifier fields of the *network join router request* inter-PAN command frame to the corresponding network parameter values as used by the initiator.

The initiator shall set the key index and encrypted network key fields of the *network join router request* inter-PAN command frame accordingly to describe the ZigBee network key to be used for securing the network.

The initiator shall set the network address field of the *network join router request* inter-PAN command frame to the selected network address with which the target shall operate on the network. If the value of the *aplFreeNwkAddrRangeBegin* attribute is equal to 0x0000, the initiator shall stochastically generate an address according to the classical ZigBee mechanism. If the value of the *aplFreeNwkAddrRangeBegin* attribute is not equal to 0x0000, the initiator shall give the target the address *aplFreeNwkAddrRangeBegin* and increment the value. The target shall not change the network address unless it leaves the network and joins another or if required to do so in order to resolve an address conflict.

If the target requested a set of group identifiers in its *scan response* inter-PAN command frame and the value of the *aplFreeGroupIDRangeBegin* attribute is not equal to 0x0000, the initiator shall allocate a range of group identifiers for the target and set the group identifiers begin and group identifiers end fields of the *network join router request* inter-PAN command frame accordingly. If the target requested a set of group identifiers in its *scan response* inter-PAN command frame and the value of the *aplFreeGroupIDRangeBegin* attribute is equal to 0x0000, the initiator shall set the group identifiers begin and group identifiers end fields of the *network join router request* inter-PAN command frame to 0x0000.

If the target indicated that it was address assignment capable in its *scan response* inter-PAN command frame and the value of the *aplFreeNwkAddrRangeBegin* attribute is not equal to 0x0000, the initiator shall allocate a range of network addresses and group identifiers that the target can use for its own purposes and set the free network address range begin, free network address range end, free group identifier range begin and free group identifier range end fields of the *network join router request* inter-PAN command frame accordingly. If the target indicated that it was address assignment capable in its *scan response* inter-PAN command frame and the value of the *aplFreeNwkAddrRangeBegin* attribute is equal to 0x0000, the initiator shall set the free network address range begin, free network address range end, free group identifier range begin and free group identifier range end fields of the *network join router request* inter-PAN command frame to 0x0000.

Once the *network join router request* inter-PAN command frame has been generated, the initiator shall transmit it to the selected target using the unicast data service. It shall then enable its receiver and wait for at *aplcRxWindowDuration* seconds or until a *network join router response* inter-PAN command frame is received with a valid transaction identifier. If a *network join router response* inter-PAN command frame with a valid transaction identifier is not received within *aplcRxWindowDuration* seconds, the initiator shall terminate the operation and perform no further processing.

On receipt of a *network join router response* inter-PAN command frame with a valid transaction identifier from the intended target, the initiator shall wait at least *aplcMinStartupDelayTime* seconds to allow the target to start operating on the network correctly.

8.4.4.2 Target procedure

On receipt of a *network join router request* inter-PAN command frame with a valid transaction identifier, the target shall decide by application specific means whether to allow itself to be joined to another network. If the target decides not to be joined to another network, it shall generate a *network join router response* inter-PAN command frame (see 7.1.2.3.4) with a status indicating failure and transmit it back to the initiator using the unicast data service; the target shall then perform no further processing. If the target decides to allow itself to be joined to another network, it shall generate a *network join router response* inter-PAN command frame with a status indicating success and transmit it back to the initiator using the unicast data service.

If the target is not factory new, it shall perform a leave request on its old network by issuing the NLME-LEAVE.request primitive to the NWK layer and wait for the corresponding NLME-LEAVE.confirm response. The target shall ensure that its old network parameters are reset to their default values, as required by the leave procedure. The target shall then copy the new network parameters to its network information base and start operating on the new network by issuing the NLME-START-ROUTER.request primitive to the NWK layer.

After the router has successfully started, it shall set *apsTrustCenterAddress* to 0xffffffffffffffff and then transmit a Device_annce command frame, as defined in section 2.4.3.1.11 of [R1] . This command frame shall be broadcast to all devices for which *macRxOnWhenIdle* is equal to True (i.e. a network address of 0xfffd).

8.4.4.3 Sequence chart for joining routers to the network

The sequence of events for joining a router target to a network (described in the previous sub-clauses) is illustrated in Figure 59.

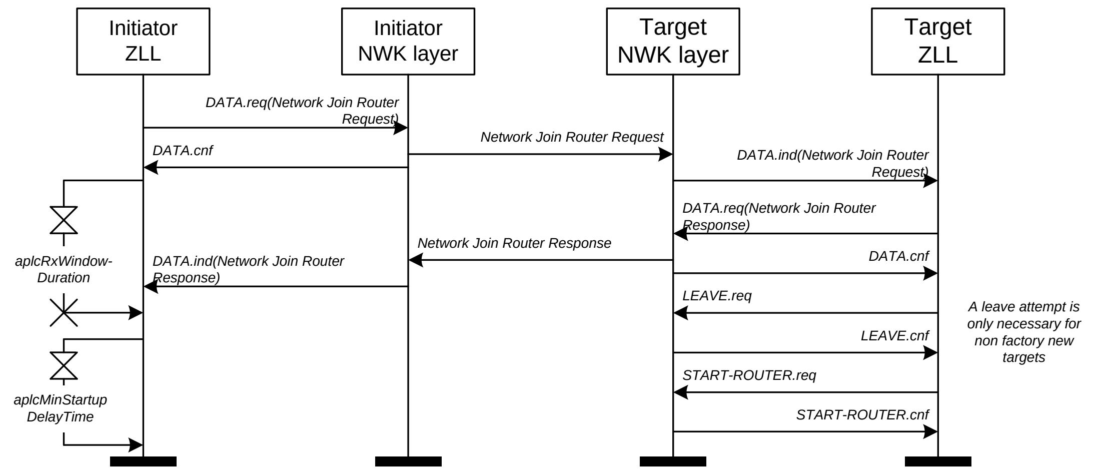

Initiator ZLL
Initiator NWK layer
Target NWK layer
Target ZLL
DATA.req(Network Join Router Request)
DATA.cnf
Network Join Router Request
DATA.ind(Network Join Router Request)
DATA.req(Network Join Router Response)
Network Join Router Response
DATA.cnf
aplcRxWindow-Duration
DATA.ind(Network Join Router Response)
LEAVE.req
LEAVE.cnf
A leave attempt is only necessary for non factory new targets
START-ROUTER.req
START-ROUTER.cnf
aplcMinStartup DelayTime

Figure 59 – Joining a router target to the network

8.4.5 Joining end devices

An end device target shall be joined to a network by an initiator that is already part of a network.

8.4.5.1 Initiator procedure

During a device discovery operation, the initiator will have found the end device target. To add the target to the network, the initiator shall generate a *network join end device request* inter-PAN command frame (see 7.1.2.2.7) as follows.

The initiator shall set the extended PAN identifier, network update identifier, logical channel and PAN identifier fields of the *network join end device request* inter-PAN command frame to the corresponding network parameter values as used by the initiator.

The initiator shall set the key index and encrypted network key fields of the *network join end device request* inter-PAN command frame accordingly to describe the ZigBee network key to be used for securing the network.

The initiator shall set the network address field of the *network join end device request* inter-PAN command frame to the selected network address with which the target shall operate on the network. If the value of the *aplFreeNwkAddrRangeBegin* attribute is equal to 0x0000, the initiator shall stochastically generate an address according to the classical ZigBee mechanism. If the value of the *aplFreeNwkAddrRangeBegin* attribute is not equal to 0x0000, the initiator shall give the target the address *aplFreeNwkAddrRangeBegin* and increment the value. The target shall not change the network address unless it leaves the network and joins another or if required to do so in order to resolve an address conflict.

If the target requested a set of group identifiers in its *scan response* inter-PAN command frame and the value of the *aplFreeGroupIDRangeBegin* attribute is not equal to 0x0000, the initiator shall allocate a range of group identifiers for the target and set the group identifiers begin and group identifiers end fields of the *network join end device request* inter-PAN command frame accordingly. If the target requested a set of group identifiers in its *scan response* inter-PAN command frame and the value of the *aplFreeGroupIDRangeBegin* attribute is equal to 0x0000, the initiator shall set the group identifiers begin and group identifiers end fields of the *network join end device request* inter-PAN command frame to 0x0000.

If the target indicated that it was address assignment capable in its *scan response* inter-PAN command frame and the value of the *aplFreeNwkAddrRangeBegin* attribute is not equal to 0x0000, the initiator shall allocate a range of network addresses and group identifiers that the target can use for its own purposes and set the free network address range begin, free network address range end, free group identifier range begin and free group identifier range end fields of the *network join end device request* inter-PAN command frame accordingly. If the target indicated that it was address assignment capable in its *scan response* inter-PAN command frame and the value of the *aplFreeNwkAddrRangeBegin* attribute is equal to 0x0000, the initiator shall set the free network address range begin, free network address range end, free group identifier range begin and free group identifier range end fields of the *network join end device request* inter-PAN command frame to 0x0000.

Once the *network join end device request* inter-PAN command frame has been generated, the initiator shall transmit it to the selected target using the unicast data service. It shall then enable its receiver and wait for at *aplcRxWindowDuration* seconds or until a *network join end device response* inter-PAN command frame is received with a valid transaction identifier. If a *network join end device response* inter-PAN command frame with a valid transaction identifier is not received within *aplcRxWindowDuration* seconds, the initiator shall terminate the operation and perform no further processing.

On receipt of a *network join end device response* inter-PAN command frame with a valid transaction identifier from the intended target, the initiator shall wait at least *aplcMinStartupDelayTime* seconds to allow the target to start operating on the network correctly.

8.4.5.2 Target procedure

On receipt of a *network join end device request* inter-PAN command frame with a valid transaction identifier, the target shall decide by application specific means whether to allow itself to be joined to another network. If the target decides not to be joined to another network, it shall generate a *network join end device response* inter-PAN command frame (see 7.1.2.3.5) with a status indicating failure and transmit it back to the initiator using the unicast data service; the target shall then perform no further processing. If the target decides to allow itself to be joined to another network, it shall generate a *network join end device response* inter-PAN command frame with a status indicating success and transmit it back to the initiator using the unicast data service.

If the target is not factory new, it shall perform a leave request on its old network by issuing the NLME-LEAVE.request primitive to the NWK layer and wait for the corresponding NLME-LEAVE.confirm response. The target shall then copy the new network parameters to its network information base and perform a network rejoin request by issuing the NLME-JOIN.request primitive to the NWK layer, ensuring the RejoinNetwork parameter is set to indicate that the device is joining the network using the NWK rejoining procedure. If the network rejoin was successful (indicated by the reception of the NLME-JOIN.confirm), the target can use the network to communicate.

After the target has successfully joined the network, it shall set *apsTrustCenterAddress* to 0xffffffffffffffff and then transmit a Device_annce command frame, as defined in section 2.4.3.1.11 of [R1] . This command frame shall be broadcast to all devices for which *macRxOnWhenIdle* is equal to True (i.e. a network address of 0xfffd).

8.4.5.3 Sequence chart for joining end devices to the network

The sequence of events for joining an end device to a network (described in the previous sub-clauses) is illustrated in Figure 60.

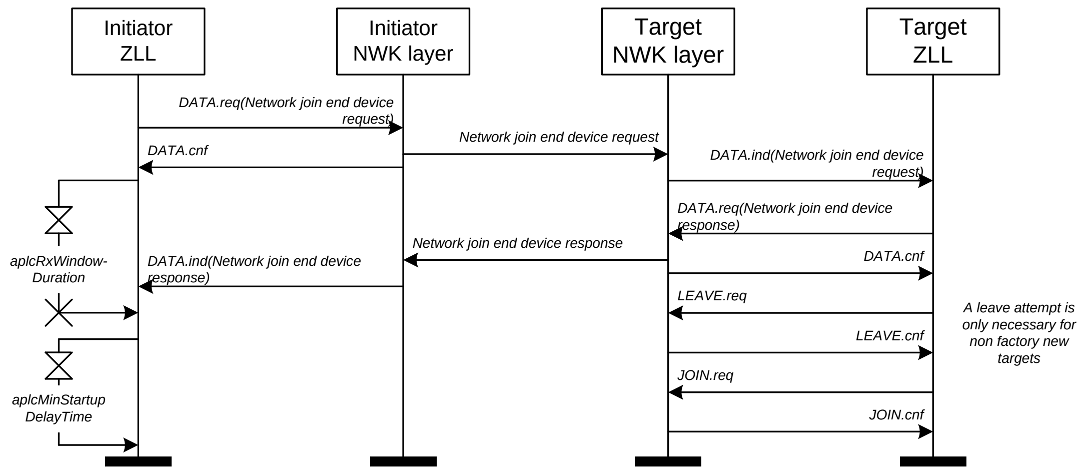

Initiator ZLL
Initiator NWK layer
Target NWK layer
Target ZLL
DATA.req(Network join end device request)
Network join end device request
DATA.cnf
DATA.ind(Network join end device request)
DATA.req(Network join end device response)
Network join end device response
aplcRxWindow-Duration
DATA.ind(Network join end device response)
DATA.cnf
LEAVE.req
A leave attempt is only necessary for non factory new targets
LEAVE.cnf
JOIN.req
aplcMinStartup DelayTime
JOIN.cnf

Figure 60 – Joining an end device target to the network

8.4.6 Network update

8.4.6.1 Initiator procedure

If an initiator finds a device during device discovery that is part of the same network as the initiator but that reports a network update identifier in its *scan response* inter-PAN command frame that is lower than that of the initiator, it may generate and transmit a *network update request* inter-PAN command frame (see 7.1.2.2.8) to the target using the unicast data service.

The *network update request* inter-PAN command frame shall contain the current network parameters of the initiator in the extended PAN identifier, network update identifier, logical channel and PAN identifier fields. In addition, the *network update request* inter-PAN command frame shall also contain the network address of the target.

Conversely, if an initiator finds a device during device discovery that is part of the same network as the initiator but that reports a network update identifier in its *scan response* inter-PAN command frame that is higher than that of the initiator, it shall update its stored network update identifier and logical channel with the values received in the *scan response* inter-PAN command frame and change to the new channel accordingly.

If the initiator is an end device, it shall then perform a network rejoin request by issuing the NLME-JOIN.request primitive to the NWK layer, ensuring the *RejoinNetwork* parameter is set to indicate that the device is joining the network using the NWK rejoining procedure. If the network rejoin was successful (indicated by the reception of the NLME-JOIN.confirm), the initiator can use the network to communicate.

8.4.6.2 Target procedure

On receipt of the *network update request* inter-PAN command frame with a valid transaction identifier (i.e. immediately following a device discovery) by a target, it shall first compare the values of the extended PAN identifier and PAN identifier fields with its corresponding stored valued. If the two values are not identical, the target shall discard the frame and perform no further processing. If the two values are identical, the target shall then compare the value of the network update identifier field with its corresponding stored value. If the value in the frame is higher than its stored value, the target shall update its stored network update identifier and logical channel with the values received in the *network update request* inter-PAN command frame, according to the policy described in 8.6. Otherwise, the target shall discard the frame and perform no further processing.

The target shall not send a response to a *network update request* inter-PAN command frame.

8.4.7 Reset to factory new

In some cases, it is desirable for a device to be removed from the network. This could happen remotely via an initiator or locally from some application stimulus.

8.4.7.1 Initiator procedure

To remove a device from its network, an initiator shall first follow the procedure for device discovery as specified in 8.4.1.1 with an extended scan of the secondary channels. Once within a transaction, the initiator shall generate and transmit a *reset to factory new request* inter-PAN command frame (see 7.1.2.2.4) to the appropriate target.

8.4.7.2 Target procedure

On receipt of a *reset to factory new request* inter-PAN command frame by a non-factory new target with a valid transaction identifier (i.e. immediately following a device discovery), it shall perform a leave request on the network by issuing the NLME-LEAVE.request primitive to the NWK layer and wait for the corresponding NLME-LEAVE.confirm response.

If the target is factory new or if a *reset to factory new request* inter-PAN command frame is received with an invalid transaction identifier (i.e. the frame was not received within the current active transaction), it shall discard the frame and perform no further processing.

The target shall not send a response to a *reset to factory new request* inter-PAN command frame.

8.4.7.3 Local device procedure

If a device receives some stimulus from the application to leave its current network, it shall perform a leave request on the network by issuing the NLME-LEAVE.request primitive to the NWK layer and wait for the corresponding NLME-LEAVE.confirm response.

If the device is factory new, it shall ignore the stimulus and perform no further processing.

8.4.7.4 Sequence chart for resetting a device to factory new

The sequence of events for resetting a device to factory new (described in the previous sub-clauses) is illustrated in Figure 61.

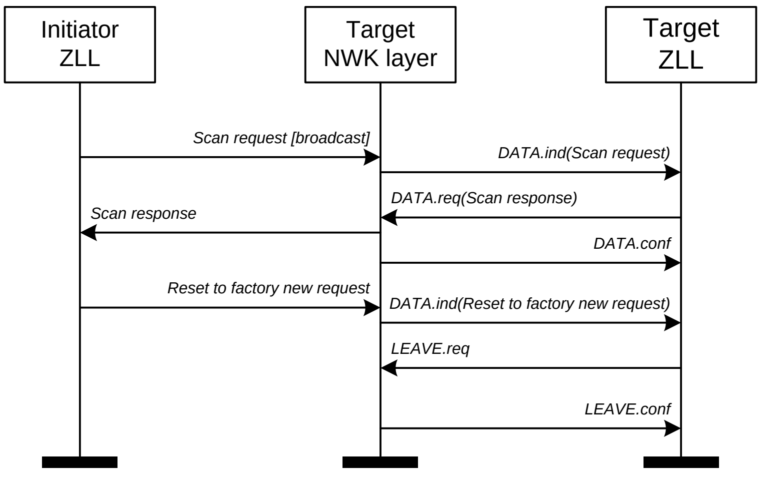


Figure 61 – Resetting a device to factory new

8.4.8 Address assignment

Network addresses and group identifiers are assigned by address assignment capable devices and all network addresses and group identifiers must be unique.

8.4.8.1 Network address assignment

Network addresses are assigned by devices that are address assignment capable. All network addresses must be unique. The method used to ensure this is to assign subdivisions of the available address space to devices that join the network and that are address assignment capable.

Since ZigBee reserves the network address 0x0000 for the coordinator and the address range (0xfff8 ⋯ 0xffff) for broadcast, the total ZLL network address space is defined in the range (0x0001 ⋯ 0xfff7). ZLL devices that are address assignment capable shall keep track of their current free network address range, $(N_{min} \cdots N_{max})$. When such a device is factory-new, N_{min} = 0x0001 and N_{max} = 0xfff7.

When a factory-new initiator device, which is address assignment capable, has just formed a new network, it shall assign itself the network address N_{min} (i.e., 0x0001) and then increment N_{min}, i.e. the range changes to (0x0002 ⋯ 0xfff7).

When a device is joined to an existing network, it shall be assigned the first (i.e. N_{min}) network address from the free network address range of the initiator through which it is joining. The initiator that started the network shall then increment N_{min}.

If a device cannot be assigned a network address, it shall not be permitted to operate on the network.

If a device that is address assignment capable joins the network, it shall also receive its own free network address range $(N'_{min} \cdots N'_{max})$. The initiator shall split its own free network address range at an implementation specified point and the upper range (i.e., highest in value) shall be assigned to the new address assignment capable device.

If after splitting the free network address range, the resulting two address ranges are smaller than an implementation specific threshold, the new device shall not be joined to the network.

8.4.8.2 Group identifier assignment

Group identifiers are used when addressing a subset of devices using broadcast mechanisms and they are typically used by a controller application residing at a certain endpoint. The group identifiers need to be unique in the network and their range is (0x0001 ⋯ 0xfeff). Group identifier 0x0000 is used for the default group in the ZCL ***scene*** cluster. Group identifiers (0xff00 ⋯ 0xffff) shall be reserved.

The number of group identifiers needed by an application residing on an endpoint is given in the device information table (see 8.1.7). Since group identifier assignment is linked to network address assignment, the total number of group identifiers needed by all endpoints on a node is reported in the *scan response* command frame (see 7.1.2.3.1). A device that is network address assignment capable shall also be group identifier assignment capable and each shall keep track of their current free group identifier range, $(G_{min} \cdots G_{max})$. When such a device is factory-new, G_{min}= 0x0001 and G_{max} = 0xfeff.

When a factory-new initiator device which is assignment capable has just formed a new network, it shall take the group identifiers, starting from G_{min}(i.e., 0x0001) for its own endpoints and shall then increment $G_{min}$with the number of endpoints supported on the device.

When a device is joined to the network, it shall receive a range of group identifiers for its endpoints and the initiator shall then increment $G_{min}$with the number of endpoints supported on the new device.

If a device that is about to be joined is also address assignment capable, it shall also receive a free group identifier range $(G'_{min} \cdots G'_{max})$, if possible. The initiator shall split its own free group identifier range at an implementation specified point and the upper range (i.e., highest in value) shall be assigned to the new address assignment capable device.

If, after division of a free group identifier range, the resulting two group identifier ranges are smaller than an implementation specific threshold, the new device shall not be joined to the network.

8.5 Classical ZigBee commissioning

8.5.1 Classical ZigBee commissioning of ZLL devices

In addition to the preferred touchlink commissioning procedure, a ZLL device shall also support the classical ZigBee commissioning mechanism, under application control, to allow it to join a non-ZLL network. Since the classical ZigBee commissioning mechanisms do not support the concept of a range of addresses, even if the device is address assignment capable it will only be allocated a single address. Network and group address ranges can only be assigned when the end device is touchlinked with another ZLL device.

If requested by the application, the device shall perform a primary network discovery by issuing the NLME-NETWORK-DISCOVERY.request primitive to the NWK layer over the primary ZLL channel set (see 8.1.2) and the *ScanDuration* parameter set to 4 (240 milliseconds scan per channel).

On receipt of the NLME-NETWORK-DISCOVERY.confirm primitive, end device shall determine whether there are any suitable networks with a permit joining flag set to TRUE. The mechanism by which the device determines whether a network is suitable is out of the scope of this specification. If a suitable network is found, the device shall attempt to join using MAC association, via the NLME-JOIN.request primitive. If the join is successful, the device shall wait to be authenticated and receive the network key from its parent. On receipt of the network key and if the device is address assignment capable, it shall set the values of the *aplFreeNwkAddrRangeBegin*, *aplFreeNwkAddrRangeEnd*, *aplFreeGroupIDRangeBegin* and *aplFreeGroupIDRangeEnd* attributes all to 0x0000. The device shall then enter its normal operating state.

If a suitable network is not found or the join was unsuccessful, the device shall perform a secondary network discovery by issuing the NLME-NETWORK-DISCOVERY.request primitive to the NWK layer over the secondary ZLL channel set (see 8.1.2) and the *ScanDuration* parameter set to 4 (240 milliseconds scan per channel).

On receipt of the NLME-NETWORK-DISCOVERY.confirm primitive, the device shall determine whether there are any suitable networks with a permit joining flag set to TRUE. The mechanism by which the device determines whether a network is suitable is out of the scope of this specification. If a suitable network is found, the device shall attempt to join using MAC association, via the NLME-JOIN.request primitive. If the join is successful, the device shall wait to be authenticated and receive the network key from its parent. On receipt of the network key and if the device is address assignment capable, the device shall set the values of the *aplFreeNwkAddrRangeBegin*, *aplFreeNwkAddrRangeEnd*, *aplFreeGroupIDRangeBegin* and *aplFreeGroupIDRangeEnd* attributes all to 0x0000. The device shall then enter its normal operating state.

If a suitable network is not found, the join was unsuccessful or the authentication failed, the device shall randomly select one of the primary ZLL channels and tune to it. If the device is an end device it shall enter a low power dormant state, performing no further processing. Conversely, if the device is a router it shall select one of the primary ZLL channels, tune to it and then enable its receiver, waiting for a touchlink command.

8.5.2 Classical ZigBee commissioning to a ZLL router

If requested by the application, a router shall enable its permit joining flag for a duration of *aplcMaxPermitJoinDuration* seconds by issuing the NLME-PERMIT-JOINING.request primitive with a *PermitDuration* of *aplcMaxPermitJoinDuration* to allow non-ZLL devices to join if required using the standard ZigBee join procedures.

If a device attempts to join using classical ZigBee commissioning, the router shall allocate an address using the classical ZigBee mechanism of generating a stochastic address.

The router shall then authenticate the device according to its current capabilities. If *apsTrustCenterAddress* is not equal to 0xffffffffffffffff, the router shall generate and transmit an APS update device command frame, secured with the trust centre link key, to the trust centre to allow it to handle the new device. If *apsTrustCenterAddress* is equal to 0xffffffffffffffff , the router shall generate and transmit an APS transport key command frame, secured with the ZLL certification pre-installed link key (see 8.7.2), to the joining device.

8.5.3 Touchlinking devices on trust centre protected networks

It is possible to perform touchlink commissioning from an initiator device to a target device connected to a trust centre protected network or between devices connected to the same trust centre protected network. A device is connected to a trust centre protected network if the value of *apsTrustCenterAddress* is not equal to 0xffffffffffffffff.

If the initiator device is connected to a trust centre protected network, devices which are either factory new or connected to another network and respond to a touchlink scan request inter-PAN command frame, shall not be joined using the touchlink procedure, i.e. the initiator device shall not transmit network start request, network join router request or network join end device request inter-PAN command frames to those devices (thus preventing devices from using touchlink commissioning to bi-pass the trust centre). However, such devices may still participate in other touchlink mechanisms such as reset to factory new or identify.

8.6 Frequency agility

ZLL supports a channel change mechanism in an application-defined way. When the channel change mechanism is instigated, the device shall broadcast a Mgmt_NWK_Update_req command frame with the *scan channels* field set to indicate the ZLL channel on which to begin operating, the *scan duration* field set to 0xfe (channel change request) and the *nwkUpdateId* field set to the value of the *nwkUpdateId* attribute of the transmitting device, incremented by one. This command frame shall be broadcast to all devices for which *macRxOnWhenIdle* is equal to True (i.e. a network address of 0xfffd).

Routers receiving this Mgmt_NWK_Update_req command frame shall update their NIB and execute their channel change procedure. End devices shall rejoin using the NWK rejoining procedure.

Routers that have missed the Mgmt_NWK_Update_req command frame can be brought back into the network through a touch-link procedure. For this reason, a device shall indicate the value of its *nwkUpdateId* attribute when it responds to a scan request via a *scan response* command frame.

If a touch-link initiator wants to bring a router back into the network (i.e. if the value of the *nwkUpdateId* indicated in the scan response command frame is older than the value of the *nwkUpdateId* attribute of the scan initiator), it shall send a unicast inter-PAN *network update request* command frame.

If a touch-link initiator detects a router reporting a *nwkUpdateId* attribute that is newer than its own *nwkUpdateId* attribute, it shall update its network settings (i.e. logical channel, PAN identifier and *nwkUpdateId*) accordingly based on the values found in the *scan response* command frame sent by that router. If the touch-link initiator is an end device, it shall execute a re-join procedure.

Note: the *nwkUpdateId* attribute can take the value 0x00 – 0xff and may wrap around so care must be taken when comparing for newness. For consistency, each device shall determine the *nwkUpdateId* to use (`ID`) using the following algorithm:

```
ID1 ← First nwkUpdateId;
ID2 ← Second nwkUpdateId;
if ( ABS( ID1 - ID2 ) > 200 ) ) then
    ID ← MIN( ID1, ID2 );
else
    ID ← MAX( ID1, ID2 );
endif
```

8.7 Security

Devices in a ZLL shall use ZigBee network layer security. Each ZLL network shall have its own network key. The network key shall be generated randomly by the initiator that starts the new network.

In this clause concatenation of strings is represented by the "||" symbol.

8.7.1 Transferring the network key during touchlink commissioning

The ZLL security architecture is based on using a fixed secret key, known as the ZLL key, which shall be stored in each ZLL device. All ZLL devices use the ZLL key to encrypt/decrypt the exchanged network key.

The architecture that is used to allow for a transfer the encrypted network key is depicted in Figure 62.

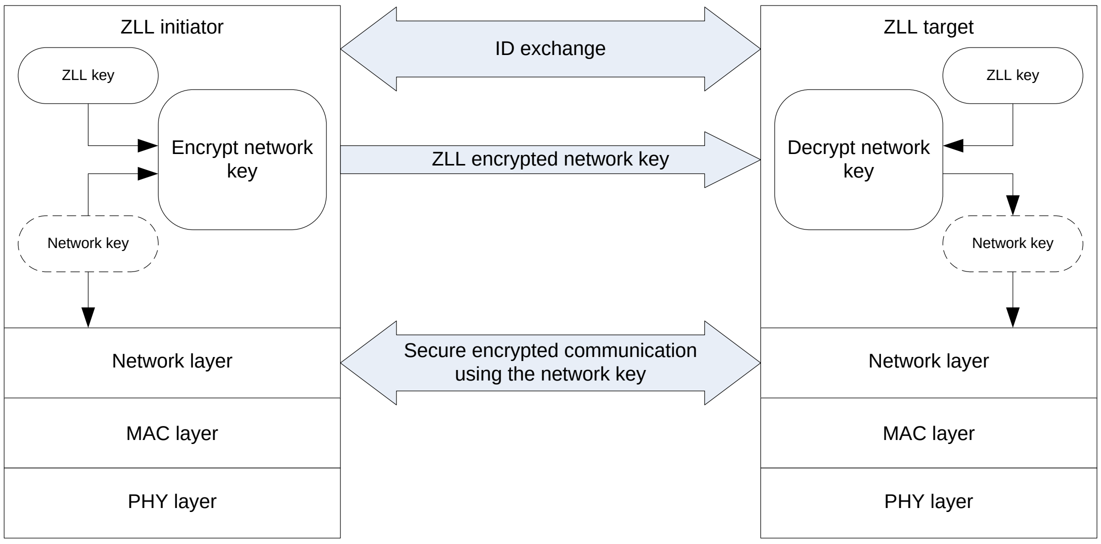


Figure 62 – Overview of ZLL security

In order to transfer the network key between the initiator and a possible target in a secure way, 16 possible algorithms can be used to encrypt the network key.

The possible target shall indicate in the key bitmask field of its *scan response* inter-PAN command frame, transmitted during device discovery, which key encryption algorithms are supported.

On receipt of each *scan response* inter-PAN command frame, the initiator shall compare the value in the received key bitmask field with its own stored key bitmask to find out if the two devices contain a common key. If no common key is found (i.e. the bitwise AND of the two is equal to zero), the initiator shall not select this target for further commissioning.

If a common key is found (i.e. the bitwise AND of the two is not equal to zero), the initiator shall set the key index to the bit position corresponding to the matching key with the highest index, encrypts the network key using the appropriate algorithm, listed in Table 67, and includes both the index and the encrypted key it in the key index and encrypted network key fields, respectively, of the *network start request*, *network join router* or *network join end device* inter-PAN command frames.

Table 67 – Key encryption algorithms

Key index	Key description	Algorithm
0	Development key	See 8.7.4
1-3	Reserved	-
4	Master key	See 8.7.5
5-14	Reserved for future use	-
15	Certification key	See 8.7.5

8.7.2 Transferring the network key during classical ZigBee commissioning

During classical ZigBee commissioning where a non-ZLL device is being joined to a ZLL network without a trust center, a pre-installed link key is used to secure the transfer of the network key when authenticating. The ZLL pre-installed link key is a secret shared by all certified ZLL devices. It will be distributed only to certified manufacturers and is bound with a safekeeping contract (see [R5]).

Prior to the successful completion of the certification, a certification pre-installed link key is used to allow testing. The certification pre-installed link key shall have the value of:

```
Certification pre-installed     0xd0 0xd1 0xd2 0xd3 0xd4 0xd5 0xd6 0xd7
link key (0:15) =               0xd8 0xd9 0xda 0xdb 0xdc 0xdd 0xde 0xdf
```

Additionally, if the decryption of the APS message fails with the key described above, ZLL devices shall try to decode the APS message using the known default trust center link key.

8.7.3 ZigBee Settings

The following ZigBee security related NIB attributes shall be set (See [ZigBee], Section 4.3.3):

- nwkSecurityLevel: 0x05 (use data encryption and frame integrity),
- nwkAllFresh: False (do not check frame counter),
- nwkSecureAllFrames: True (only accept secured frames).

8.7.4 Key index 0

The network key encryption algorithm with a key index equal to 0 is known as the development key. This algorithm encrypts the network key with AES in ECB mode in one single step where the AES key is equal to:

```
"PhLi" || TrID || "CLSN" || RsID
```

Where `TrID` is the transaction identifier field of the original *scan request* command frame passed between the initiator and target and `RsID` is the response identifier of the *scan response* command frame passed between the target and the initiator (both values are random 32-bit integers). The ASCII characters in quotes ("") should be converted to their equivalent hexadecimal byte values, with the leftmost character being the leftmost byte.

For example:

Encrypted Network Key (0:15)	0x48 0x3c 0x2b 0x19 0x7c 0x27 0xc3 0xcc 0x76 0xa3 0xd6 0x3b 0x2e 0xa8 0xdb 0x0b
Transaction identifier	0xea9cd138
Response identifier	0x8f8dbab4
Resulting AES Key (0:15)	0x50 0x68 0x4c 0x69 0xea 0x9c 0xd1 0x38 0x43 0x4c 0x53 0x4e 0x8f 0x8d 0xba 0xb4
Decrypted Network Key (0:15)	0xac 0xbe 0xf1 0x44 0x70 0x27 0xd8 0xd9 0x5a 0xfa 0x42 0xb0 0x77 0xe4 0x88 0xa5

Note: The development key (key index 0) shall only be used during the development phase of Light Link products. Commercial Light Link products shall not use nor indicate having support for the development key.

8.7.5 Key index 4 and 15

8.7.5.1 Key usage

The ZLL security details described in this section apply to key index 4 and 15 of Table 67. The secure NWK key transport methods indicted by key index 4 and 15 use the same algorithm, as described in section 8.7.5.2. However, they differ in the type of ZLL Key they use for NWK key protection.

8.7.5.1.1 Master key (key index 4)

The ZLL master key is a secret shared by all certified ZLL devices. It will be distributed only to certified manufacturers and is bound with a safekeeping contract (see [R5]).

The ZLL device using the master key in combination with the algorithm described in this document shall always set bit 4 in the key bitmask field of the *scan response* command frame to 0b1 (see 7.1.2.3.1).

8.7.5.1.2 Certification key (key index 15)

Prior to the successful completion of the certification, a certification key is used to allow testing of the security mechanisms as specified in this document. The certification key shall have the value of:

```
    Certification key (0:15) =   0xc0 0xc1 0xc2 0xc3 0xc4 0xc5 0xc6 0xc7
                                 0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf
```

The ZLL device using the certification key in combination with the algorithm described in this document shall always set bit 15 in the key bitmask field of the *scan response* command frame to 0b1 (see 7.1.2.3.1).

The certification key may also be used during the development phase of Light Link products. However, commercial Light Link products shall not use nor indicate having support for the certification key.

8.7.5.2 Algorithm

8.7.5.2.1 Encrypting network keys for ZLL initiator

The ZLL touch-link initiator shall perform the following steps to encrypt the network key and transport it to the ZLL target:

- Exchange of transaction identifier and response identifier as part of the touch-linking procedure.
- Derive the ephemeral transport key (see Figure 63) from the transaction identifier, response identifier and ZLL master or certification key, as described in 8.7.5.2.3.
- Encrypt the ZLL network key using the calculated transport key and the AES ECB mode, as described in 8.7.5.2.3.
- Transmit the encrypted network key to the ZLL target as part of the touch-linking procedure, as described in 8.4.

8.7.5.2.2 Decrypting network keys for ZLL target

The ZLL touch-link target shall perform the following steps to decrypt the network key received from the ZLL touch-link initiator:

- Exchange of transaction identifier and response identifier as part of the touch-linking procedure.
- Receive the encrypted network key as part of the touch-linking procedure, as described in 8.4.

- Derive the transport key (see Figure 63) from the transaction identifier, response identifier and ZLL master or certification key, as described in 8.7.5.2.3.
- Decrypt the received encrypted network key by using the calculated transport key and the AES ECB mode, as described in 8.7.5.2.3.
- Store the received network key in the NIB parameter of the ZLL target.

8.7.5.2.3 Calculations required for the encryption/decryption of the network key

The encryption/decryption key calculation to encrypt/decrypt the network key is illustrated in Figure 63.

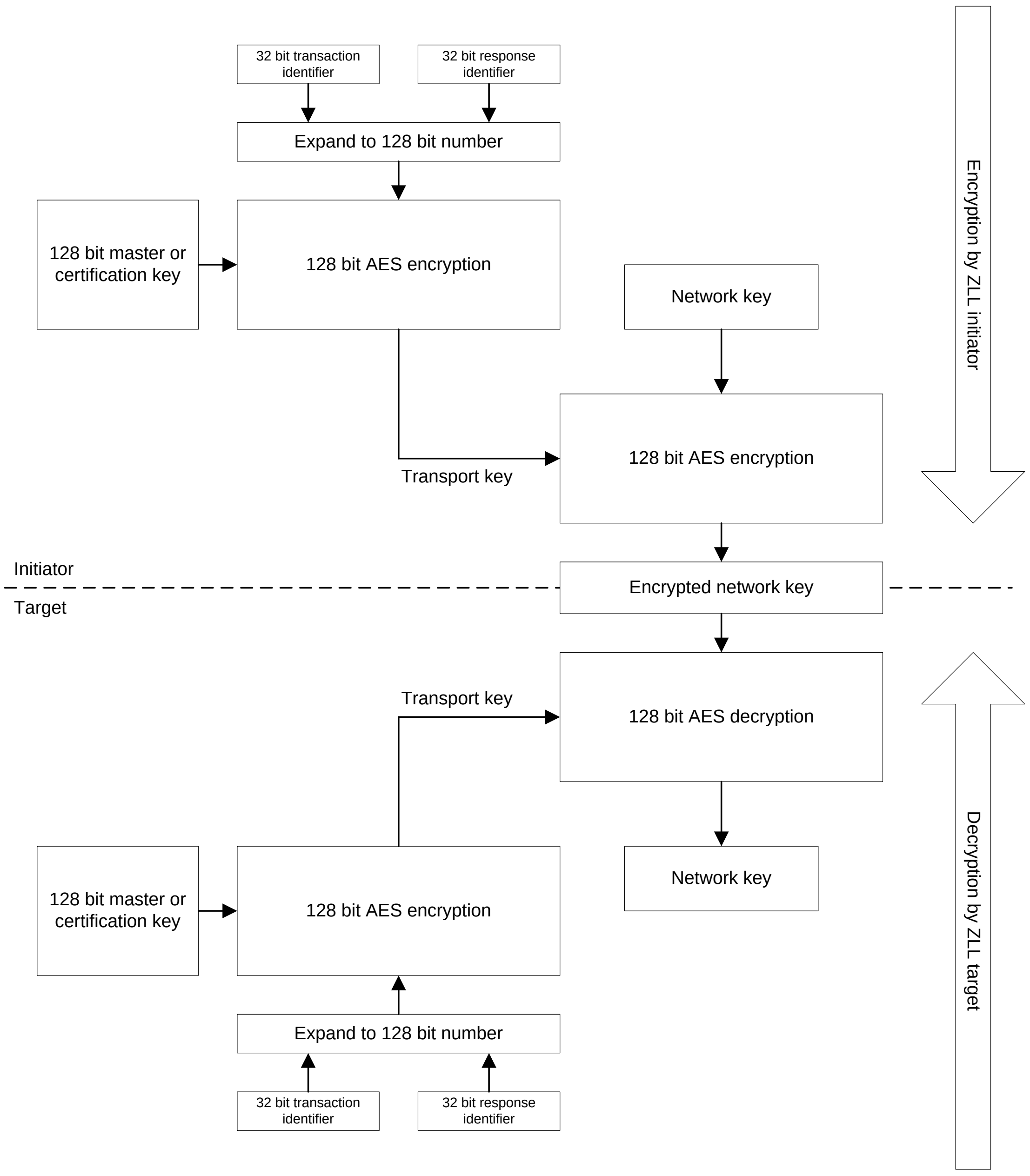


Figure 63 – Steps required to encrypt/decrypt the network key

Unless explicitly specified otherwise, all numbers in this chapter are formatted little Endian, i.e. with their least significant octet first.

The basic ingredients to perform the encryption/decryption of the network key are:

- The 32 bit transaction identifier
- The 32 bit response identifier
- The ZLL master or certification key

The encryption of the network key is performed by the following processing steps:

A. Merge and expand the transaction identifier and response identifier into a 128 bit number by concatenating them (in Little Endian representation) as follows:

   Transaction identifier || transaction identifier || response identifier || response identifier.

B. Calculate the transport key by executing the 128 bit AES encryption with the expanded 128 bit number obtained from step A as *plaintext*, and ZLL master or certification key as *key*.

C. Encrypt the network key by executing the 128 bit AES encryption using the network key as *plaintext* and the transport key obtained from step B as *key*.

The decryption of the network key is performed by the following processing steps:

D. Merge and expand the transaction identifier and response identifier into a 128 bit number, as described in step A above.

E. Calculate the transport key by executing the 128 bit AES encryption with the expanded 128 bit number obtained from step D used as *plaintext*, and ZLL master or certification key as *key*.

F. Decrypt the network key by executing the 128 bit AES decryption with the transport key obtained from step E as *key* and the encrypted network key as *ciphertext*.

All AES functions used in steps B, C and E above shall use AES encryption in ECB mode and the AES function in step F shall use AES decryption in ECB mode [R7] .

9 Annex A: ZLL security test vectors

This annex provides sample test vectors for the ZLL security specification (as defined in sub-clause 8.7), in order to assist in building interoperable security implementations.

9.1 ZLL Initiator operation

ZLL Certification Key (0:15)	`0xc0 0xc1 0xc2 0xc3 0xc4 0xc5 0xc6 0xc7 0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf`
Transaction ID	`0x3eaa2009`
Response ID	`0x88762fb1`
Expanded input (0:15)	`0x3e 0xaa 0x20 0x09 0x3e 0xaa 0x20 0x09 0x88 0x76 0x2f 0xb1 0x88 0x76 0x2f 0xb1`

After AES ECB encryption:

Transport Key (0:15)	`0x66 0x9e 0x08 0xe4 0x02 0x77 0xed 0x9a 0xb3 0x6b 0x25 0x80 0x45 0x6b 0x41 0x76`
NWK key (0:15)	`0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xaa 0xbb 0xcc 0xdd 0xee 0xff 0x00`

After AES ECB encryption:

Encrypted Network Key (0:15)	`0x83 0x22 0x63 0x68 0x73 0xa7 0xbb 0x2a 0x18 0x9a 0x53 0x70 0x8c 0x60 0x7b 0xd0`

ZLL Target operation

ZLL Certification Key (0:15)	`0xc0 0xc1 0xc2 0xc3 0xc4 0xc5 0xc6 0xc7 0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf`
Transaction ID	`0x3eaa2009`
Response ID	`0x88762fb1`
Expanded input (0:15)	`0x3e 0xaa 0x20 0x09 0x3e 0xaa 0x20 0x09 0x88 0x76 0x2f 0xb1 0x88 0x76 0x2f 0xb1`

After AES ECB encryption:

Transport Key (0:15)	`0x66 0x9e 0x08 0xe4 0x02 0x77 0xed 0x9a 0xb3 0x6b 0x25 0x80 0x45 0x6b 0x41 0x76`
Received encrypted NWK key (0:15)	`0x83 0x22 0x63 0x68 0x73 0xa7 0xbb 0x2a 0x18 0x9a 0x53 0x70 0x8c 0x60 0x7b 0xd0`

After AES ECB decryption:

NWK key (0:15)	`0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xaa 0xbb 0xcc 0xdd 0xee 0xff 0x00`

Note: the first (i.e., leftmost on the page) byte of the encrypted network key is sent first in the associated encrypted network key fields of the network start request, network join router request and network join end device request inter-PAN command frames.