zigbee alliance

Zigbee Cluster Library Specification

Chapter Document 14-0125

Cluster Library 07-5123 Revision 7

Zigbee Document	075123
Date of release	Feb 2018
Sponsored by	Zigbee Alliance
Accepted by	This document has been accepted for release by the Zigbee Alliance Board of Directors.
Abstract	This document defines the Zigbee Cluster Library.
Keywords	Zigbee, Application Layer, Data Model, Cluster Library, ZCL, Dotdot Catalog

Notice of Use and Disclosure

Participants

The following is a list of Zigbee members who contributed to this document:

Cam Williams - Chair of the Foundation Working Group and ZCL Editor

Rob Alexander	Jared Lemke
Shane Almeida	Christopher Leidigh
Casey Anderson	Yingbo Li
Skip Ashton	Marco Naeve
Wally Barnum	Juan Agui Martin
Alex Chu	Christian P. Garcia
Ettore Colicchio	Jeff Mathews
Jeff Cooper	Tony Mauro
Damon Corbin	Leslie Mulder
John Cowburn	Luca Negri
Robert Cragie	Ivan O’Neill
Jonathan Cressman	Isaac Pinhas
Tim Gillman	Andrea Ranalli
Drew Gislason	Jonas Riska
Ezra Hale	Zachary Smith
Jesper Haee	Robby Simpson
Robert Hall	Sumit Singh
Jon Harros	David Smith
Jim Hartman	Matt Smith
Arasch Honarbacht	Michael Stuber
Ted Humpal	Don Sturek
Phil Jamieson	Mads Westergreen
William Keith	Urban Wicklander
Larry Kohrmann	Cam Williams
Tom Klein	Ian Winterburn
John Knuth	Kenny York
Cristian Kuster	Walter Young
Zin Kyaw	Juan Agui Martin
Gary Lee	Jeff Mathews

Document Control

The Zigbee Cluster Library is made of individual chapters such as this one. See Chapter 1 for the list of all chapters. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

An update to any of these chapters will be reflected in an update to the source document list below.

Chapter 1 – Introduction	Document 14-0125-13
Chapter 2 – Foundation	Document 14-0126-16
Chapter 3 – General	Document 14-0127-20
Chapter 4 – Measurement and Sensing	Document 14-0128-11
Chapter 5 – Lighting	Document 14-0129-15
Chapter 6 – HVAC	Document 14-0130-12
Chapter 7 – Closures	Document 14-0131-15
Chapter 8 – Security and Safety	Document 14-0132-13
Chapter 9 – Protocol Interfaces	Document 14-0133-08
Chapter 10 – Smart Energy	Document 14-0134-11
Chapter 11 – Over the Air Upgrades	Document 14-0135-15
Chapter 12 – Telecommunications	Document 14-0136-10
Chapter 13 – Commissioning	Document 14-0137-13
Chapter 14 – Retail Services	Document 14-0138-08
Chapter 15 – Appliances	Document 14-0139-12
Approved Errata for this ZCL revision	Document 17-2017
Source files for drawings in this ZCL revision	Document 14-0141-00

46 # Document History

Rev	Date	Comments
00	11-Jul-2007	Document created
01	19-Oct-2007	First release
02	29-May-2008	Added Commissioning Cluster from 064699r12. • Added material from annex of CBA Profile 053516r10 • Structured types (arrays etc) and structured R/W commands • Input / Output / Value clusters (Basic) • Input / Output / Value clusters (BACnet Regular & Extended) • Generic Tunnel cluster • BACnet Protocol Tunnel cluster Made changes to the Color Control cluster re. CCB 870 • Added x,y control according to CIE 1931 Color Space Added long data types (as required by SE profile 075356r12 etc) • 40-64bit integers etc, long strings Made changes to time cluster (as required by CCBs 890, 914) • Added time zone & DST + UTCtime type Made minor changes as requested by the following CCBs • 627, 714, 781, 853, 854, 867, 878, 879, 880, 881, 883, 893, 897, 898, 919, 958
03	18-Sep-2009	The following changes were made to the Editor's Copy of the ZCL, 095254r00. Made change to the Basic cluster, re CCB comment #606 • Added optional attribute *DisableLocalConfig*. Updated Pressure Measurement cluster re CCB comment #961 • Added extra attributes to allow wider range of pressure. Updated Color Control cluster re CCB comment #1006 • Clarification of stop commands, color mode switching etc. Made changes to RSSI Location cluster, re CCB comment #1053 • Added mechanism for centralized location. Made change to Generic Tunnel cluster, re CCB comment #1068 • Added extra fields to Match Protocol Address Response Command
	24-Dec-2009	Made minor changes and clarifications re the following CCBs • 960, 1001, 1004, 1061, 1097. • Added Door Lock cluster. Updated Occupancy Sensor re CCB comments 1092, 1093, 1094
04	2010	CCB 1174: Fixed references CCB 1176: Added new status codes CCB 1202: Corrected default value in thermostat cluster
	Apr-2012	CCB 1381: Default Response clarification

Rev	Date	Comments
		CCB 1260: Generic Tune l cluster clarification CCB 1377: Commissioning Cluster minor change CCB 1146: Report Attributes without Configuration CCB 1169: Dependencies on Optional Attributes CCB 1379: Generic Tunnel *ProtocolAddress* attribute ReadOnly Option CCB 1420: Time cluster ESI bit CCB 1390: Reporting destination clarification
05	18-Mar-2015	Move to individual chapters Added all approved Clusters from other Application Specifications Included CCBs Editorial cleanup of document
06	14-Jan-2016	Chapter 1: New terms for Zigbee 3.0 Chapter 2: Zigbee 3.0 & Application Architecture changes Broadcast Endpoint Rules Global discovery commands from ZHA 1.2 CCB 1277 1319 1444 1485 1505 1578 1923 2029 2092 Chapter 3: *ZCLVersion* attribute of Basic cluster is 0x02 CCB 1480 1555 1647 1745 1809 1815 1822 1833 2100 Chapter 4: CCB 2048 2049 2050 2055 Chapter 5: ZLL 1.0 errata CCB 2028 2106 Chapter 6: CCB 1485 1823 Chapter 7: CCB 1811 1812 1821 1994 1995 1996 1997 2086 2094 2095 2096 2097 Chapter 8: ZHA 1.2 & 1.2.1 & errata CCB 1977 2044 2045 Chapter 11: CCB 1374 1470 1477 1540 1594 2046 2056 Chapter 15: CCB 1893
07	Jan-2018	Removed the extraneous word “ZigBee” to describe items. CCB 2288 Chapter 1: reference for Manufacture Code database Chapter 2: clarified cluster Instance Model CCB 2327 2266 2338 2213 2318 Define Deprecation New data type: Fixed ASCII Chapter 3: Level Control cluster State Change Table New Basic attributes; ZCLVersion is 0x03 Transition time to Recall Scene NFR Quality of Goods clusters: PWM, Level ZLO 1.0 changes to Level Control for Lighting CCB 1499 1584 1775 2085 2147 2197 2211 2212 2229 2281 2289 CCB 2329 2330 2333 2309 2319 Chapter 4: NFR Quality of Goods Measurement clusters: Wind Speed, Concentration, pH, Electrical Conductivity

Rev	Date	Comments
		Physical Contact Occupancy CCB 2167 2236 2241 2370 Chapter 5: ZLO 1.0; *Options* Attribute; CCB 2085 2104 2124 2193 2230 2393 Deprecate some attributes Chapter 6: CCB 1981 2186 2249 2250 2251 Thermostat Setback Chapter 7: CCB 2328 2340 2316 Chapter 8: CCB 2341 2350 2352 Door-Window Position feature Chapter 10: CCB 2288 CCB 2339 Chapter 11: alternative Image Activation Policies; 128-bit Crypto suite CCB 2019 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 CCB 2296 2307 2315 2339 2342 2398 2464 Chapter 13: Touchlink Profile Interop bit; CCB 2115 2105 Chapter 15: Cleaned up ranges to follow reserved value rules

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

The Zigbee Cluster Library is made of individual chapters such as this one. See Document Control in the Zigbee Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

1.1 Scope and Purpose

This document specifies the Zigbee Cluster Library (ZCL). The ZCL is a repository for cluster functionality that is developed by the Zigbee Alliance, and is a working library with regular updates as new functionality is added.

A developer constructing a new application should use the ZCL to find relevant cluster functionality that can be incorporated into the new application. Correspondingly, new clusters that are defined for applications should be considered for inclusion in the ZCL.

The ZCL consists of the ZCL Foundation, a set of elements that apply across the entire library (such as frame structures, attribute access commands and data types), and a number of sets of clusters. Clusters that are generally useful across many application domains are included in the General set. Clusters that are intended for use mainly in specific application domains are grouped together in domain oriented sets.

1.2 Acronyms and Abbreviations

Table 1-1. Acronyms and Abbreviations

Acronym	**Definition**
Acc	Access
ACE	Ancillary Control Equipment
AES	Advanced Encryption Standard
AIB	Application support sub-layer Information Base
AMI	Advanced Metering Infrastructure *or* Advanced Metering Initiative
AP	Access Point
APS	Application support Sub-layer
BPL	Broadband over Power Lines
CA	Certificate Authority
CBA	Commercial Building Automation
CBKE	Certificate-based Key Establishment
CIE	Control and Indicating Equipment
CT	Commissioning Tool
D	Deprecated
Def	Default
ECDSA	Elliptic Curve Digital Signature Algorithm

Acronym	Definition
ECMQV	Elliptic Curve Menezes-Qu-Vanstone
EFT	Electronic Funds Transfer
EMS	Energy Management System
EOF	End of File
EPID	Extended PAN Identifier
ESI	Energy Service Interface
ESP	Energy Service Portal
EUI64	Extended Universal Identifier-64
GPRS	General Packet Radio Service
HA	Home Automation (Application Profile)
HAN	Home Area Network
HVAC	Heating, Ventilation, Air Conditioning
IAS	Intruder Alarm System
ID	Information delivery
ID	Identifier (or Id)
IHD	In-Home Display
IN	Information node
IPD	In-Premises Display (Same as IHD) or Inter-PAN Device
IVR	Interactive Voice Response
M	Mandatory
M/O	Mandatory or Optional
MAC	Medium Access Control (referring to protocol stack sublayer)
MAC	Message Authentication Code (referring to cryptographic operation)
MAC PIB	Medium Access Control sub-layer PAN Information Base
m-commerce	Mobile commerce
MRD	Market Requirements Document
MT	Mobile Terminal
NAN	Neighborhood Area Network
NIB	Network layer Information Base
NWK	Network layer
O	Optional
OTA	Over the Air
P	Mandates that an attribute is reportable
P2P	Peer to Peer

Acronym	**Definition**
PAN	Personal Area Network
PCT	Programmable Communicating Thermostat
PD	Payment Device
PHHC	Personal Home and hospital Health Care
PID	PAN Identifier
PIR	Pyroelectric Infra-Red (a type of motion detection sensor)
PKKE	Public Key Key Establishment
POS	Point of Sales
R	Readable (Read) or Read only if not also designated as Writable (W)
R*W	Readable and optionally writable
RAN	RSSI Anchor Node
RFD	Reduced Functionality Device
RLG	RSSI Location Gateway
RLN	RSSI Location node
RSSI	Received Signal Strength Indication
R/W	Readable and Writable (same as RW)
S	Mandates that an attribute is part of a scene, if the Scene cluster is on the same endpoint
SAS	Startup Attribute Set
SE	Smart Energy (Application Profile)
SED	Sleepy End Device is a Zigbee End Device with rxOnWhenIdle set to FALSE
SKKE	Symmetric Key Key Exchange
SVE	Specification Validation Event
TC	Trust Center
TOU	Time of Use
TRD	Technical Requirements Document
UKE	Unprotected Key Establishment
UTF-8	8-bit Unicode Transformation Format
W	Writable (Write) or Write only if not also designated as Readable
WD	Warning Device
ZCL	Zigbee Cluster Library
ZCL*n*	A revision of the ZCL. For example: ZCL6 is the Zigbee Cluster Library revision 6
ZDO	Zigbee Device Object
ZDP	Zigbee Device Profile

Acronym	Definition
ZED	Zigbee End Device (equivalent to IEEE's RFD – Reduced Functionality Device)
ZR	Zigbee Router (equivalent to IEEE's FFD – Full Functionality Device)

1.3 Definitions

Many of these terms are described in more detail in the core stack specification [Z1], or the Application Architecture specification [Z5].

Application Cluster: An application cluster generates persistent functional application transactions between client and server.

Attribute: A data entity which represents a physical quantity or state. This data is communicated to other devices using commands.

Binding: A persistent mapping of a local cluster instance to one or more corresponding remote cluster instances. A binding can be broadcast, groupcast, or unicast. A unicast binding includes an address (IEEE or network) and endpoint.

Cluster: A cluster is a specification defining one or more attributes, commands, behaviors and dependencies, that supports an independent utility or application function. The term may also be used for an implementation or instance of such a specification on an endpoint.

Cluster identifier: The cluster identifier is a 16-bit number that maps to (identifies) a single cluster specification. More than one cluster identifier may map to a cluster specification, each defining a different scope and purpose. Cluster identifiers are designated as inputs or outputs in the simple descriptor for use in creating a binding table.

Client: A cluster interface which is listed in the output cluster list of the simple descriptor on an endpoint. Typically this interface sends commands that manipulate the attributes on the corresponding server cluster. A client cluster communicates with a corresponding remote server cluster with the same cluster identifier.

Corresponding cluster: The opposite side of a cluster (client to a server, or server to a client).

Device: A specification which defines a unique device identifier and a set of mandatory and optional clusters to be implemented on a single endpoint. The term may also be used for an implementation or instance of the device specification on an endpoint.

Node: A Zigbee node (or node) is a single testable implementation of a Zigbee application on a single Zigbee stack, with a single network address, on a single network.

Product: A product is a node that is intended to be marketed.

Server: A cluster interface which is listed in the input cluster list of the simple descriptor on an endpoint. Typically, this interface supports all or most of the attributes of the cluster. A server cluster communicates with a corresponding remote client cluster with the same cluster identifier.

Service discovery: The ability of a device to locate services of interest.

Sleepy End Device: A Zigbee End Device with rxOnWhenIdle set to FALSE.

Utility Cluster: A utility cluster is not part of the application function of the product. It may be used for commissioning, configuration, discovery, addressing, diagnostics, etc.

Type 1 Cluster: A type 1 cluster's primary function is to initiate transactions from the client to the server.

Type 2 Cluster: A type 2 cluster's primary function is to initiate transactions from the server to the client.

Zigbee Coordinator: An IEEE 802.15.4-2003 PAN coordinator.

Zigbee End Device: an IEEE 802.15.4-2003 RFD or FFD participating in a Zigbee network, which is neither the Zigbee coordinator nor a Zigbee router.

Zigbee Router: an IEEE 802.15.4-2003 FFD participating in a Zigbee network, which is not the Zigbee coordinator but may act as an IEEE 802.15.4-2003 coordinator within its personal operating space, that is capable of routing messages between devices and supporting associations.

1.4 Conformance Levels

Expected: A key word used to describe the behavior of the hardware or software in the design models *assumed* by this Draft. Other hardware and software design models may also be implemented.

May: A key word that indicates flexibility of choice with *no implied preference*.

Shall: A key word indicating a mandatory requirement. Designers are *required* to implement all such mandatory requirements.

Should: A key word indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase *is recommended*.

1.5 References

The following standards and specifications contain provisions, which through reference in this document constitute provisions of this specification. All the standards and specifications listed are normative references. At the time of publication, the editions indicated were valid. All standards and specifications are subject to revision, and parties to agreements based on this specification are encouraged to investigate the possibility of applying the most recent editions of the standards and specifications indicated below.

1.5.1 Zigbee Alliance Documents

[Z1]	Zigbee 053474, Zigbee Specification
[Z2]	Zigbee 064321, Zigbee Stack Profile
[Z3]	Zigbee 074855, Zigbee PRO Stack Profile
[Z4]	Zigbee 08006, Zigbee-2007 Layer PICs and Stack Profiles
[Z5]	Zigbee 130589, Application Architecture
[Z6]	Zigbee 130402, Base Device Behavior Specification
[Z7]	Zigbee 053298, Profile Identifier Database
[Z8]	Zigbee 106050, Zigbee Device internetworking, list of Device IDs
[Z9]	Zigbee 075356, Smart Energy Profile Specification
[Z10]	Zigbee 03084, Zigbee Key Establishment Proposal
[Z11]	Zigbee 095343, Installation Code Sample Source Code
[Z12]	Zigbee 053874, Manufacturer Code Database
[Z13]	Zigbee 115456, Master Cluster List
[Z14]	Zigbee 1602867, Zigbee 3.0 Cluster List

1.5.2 International Standards Documents

[I1]	CIE 1931 Color Space. Commission Internationale de l'Eclairage Proceedings. Cambridge University Press, Cambridge
[I2]	ISO 7816 International Standard for Electronic Identification Cards with Contacts (Smart Cards)

1.5.3 National Standards Documents

[N1] EN 50131 European Standards Series for Intruder Alarm Systems

[N2] BSI British Standards, document BS EN 50523-2:2009, "Household Appliances interworking – Part 2: Data Structures". July 2009

[N3] NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: CCM Mode for Authentication and Confidentiality, May 2004

[N4] FIPS Pub 197, Advanced Encryption Standard (AES), Processing Standards Publication 197, US Department of Commerce/NIST Springfield, Virginia, November 26, 2001

[N5] FIPS Pub 198, The Keyed-Hash Message Authentication Code (HMAC), Federal Information, Processing Standards Publication 198, US Department of Commerce/NIST Springfield, Virginia, March 6, 2002

1.5.4 IEEE Documents

[E1] IEEE Standards 802, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), IEEE, October 2003.

[E2] IEEE 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, IEEE, 1985.

1.5.5 ASHRAE Documents

[A1] ASHRAE 135-2004 standard, Data Communication Protocol for Building Automation and Control Networks

1.5.6 Health Care Documents

[H1] ISO/IEEE 11073-20601: Health Informatics - Personal Health Device Communication - Application Profile - Optimized Exchange Protocol - version 1.0 or later.

[H2] ISO/IEEE P11073-10404, Health informatics – Personal health device communication – Device specialization – Pulse oximeter.

[H3] ISO/IEEE P11073-10407, Health informatics – Personal health device communication – Device specialization – Blood pressure monitor.

[H4] ISO/IEEE P11073-10408, Health informatics – Personal health device communication – Device specialization – Thermometer.

[H5] ISO/IEEE P11073-10415, Health informatics – Personal health device communication – Device specialization – Weighing scale.

[H6] ISO/IEEE P11073-10417, Health informatics – Personal health device communication – Device specialization – Glucose meter.

[H7] ISO/IEEE P11073-10419, Health informatics – Personal health device communication – Device specialization – Insulin Pump

[H8] ISO/IEEE P11073-10421, Health informatics – Personal health device communication – Device specialization – Peak Expiratory Flow Monitor

[H9] ISO/IEEE P11073-10441, Health informatics – Personal health device communication – Device specialization – Cardiovascular Fitness and Activity Monitor.

[H10] ISO/IEEE P11073-10442, Health informatics – Personal health device communication – Device specialization – Strength Fitness Equipment.

[H11] ISO/IEEE P11073-10471, Health informatics – Personal health device communication – Device specialization – Independent living activity hub.

[H12] ISO/IEEE P11073-10472, Health informatics – Personal health device communication – Device specialization – Medication Monitor.

1.5.7 Other Documents

[O1] Standards for Efficient Cryptography: SEC 1 (working draft) ver 1.7: Elliptic Curve Cryptography, Certicom Research, www.secg.org, November 13, 2006

[O2] Standards for Efficient Cryptography: SEC 4 (draft) ver 1.0: Elliptic Curve Cryptography, Certicom Research, www.secg.org, January 24, 2013

[O3] RFC 3280: Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile, IETF, www.ietf.org, April 2002

1.6 Conventions

The following conventions are used in this document.

1.6.1 Enumerations and Reserved Values

Each undefined value or range of an enumeration, field, or identifier SHALL be considered reserved for future revisions of this standard and SHALL not be available for implementation.

Each value or range of an enumeration, field, or identifier that is available for non-standard implementation SHALL be described as “manufacturer specific”, “vendor specific”, “ms”, or “MS”.

Each value or range of an enumeration, field, or identifier that is available for other parts of this standard SHALL be described as such.

Each value or range of an enumeration, field, or identifier that is obsolete, and not available for implementation, SHALL be described as “Obsolete”.

1.6.2 Reserved Bit Fields

Each full or partial data field (e.g., message data field), of any bit length, that is undefined, SHALL be considered reserved for future revisions of this standard and SHALL not be available for implementation.

Please see Section Chapter 2, Transmission and Reception, regarding rules for setting and interpreting reserved fields.

1.6.3 Number Format

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.

Binary numbers are specified as successive groups of 4 bits, separated by a space (“ “) character from the most significant bit (next to the 0b prefix and left most on the page) to the least significant bit (rightmost on the page), e.g. the binary number 0b0000 1111 represents the decimal number 15. Where individual bits are indicated (e.g. bit 3) the bit numbers are relative to the least significant bit which is bit 0.

When a bit is specified as having a value of either 0 or 1 it is specified with an “x”, e.g. “0b0000 0xxx” indicates that the lower 3 bits can take any value but the upper 5 bits must each be set to 0.

1.7 Testing, Validation and Certification

The text of this document has been balloted and approved according to the procedures existing at the time of each release. However, not all cluster specifications have undergone a complete and successful specification validation event (SVE), which requires the testing of multiple diverse implementations to verify a common interpretation of the text.

New features may be added to new releases of this document, but only if the specification text has passed an SVE. Old releases did not follow this rule, hence the statement above. Even a successful SVE for a cluster specification may not test or validate all cluster specification text.

This is a living document. Erroneous or ambiguous text is discovered regularly, reviewed, approved and added as errata to this document. Testing and validation of errata may lag the release of errata text.

Not all clusters in this specification are certifiable under a current certification program. A certification program defines a testing process that may require reference implementations, test fixtures, and/or test scripts. A certification program may also be frozen or inactive. A frozen certification program does not support new errata or new features. If there is no active certification program or the testing process requirements are not in place to test a cluster specification, then the cluster specification is not able to be certified.

As of the writing of this section, the active and unfrozen certification programs for this specification are for Zigbee 3.0 and Smart Energy. Please see documents [Z13] and [Z14] which track the status of criteria for cluster specification certification

CHAPTER 2 FOUNDATION

The Zigbee Cluster Library is made of individual chapters such as this one. See Document Control in the Zigbee Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

2.1 Scope and Purpose

This chapter provides an entry point into the documentation for the Zigbee Cluster Library (ZCL), and specifies the elements that are general across the entire library.

The ZCL frame structure is specified along with ZCL wide commands used to manipulate attributes from all the clusters defined throughout the ZCL. In addition, a set of data types is defined that can be used to represent attributes and a common set of status values returned by commands throughout the ZCL.

An overview is included which lists all the domains specified in the ZCL and the clusters contained therein.

2.2 Cluster Library Overview

The Zigbee Cluster Library (ZCL) is intended to act as a repository for cluster functionality and it is a working library with regular updates as new functionality is added. A developer constructing a new application SHOULD use the ZCL to find relevant cluster functionality that can be incorporated into the new application so as not to “re-invent the wheel”. This also allows applications to be developed with more of an object-oriented style approach.

2.2.1 Architecture and Data Model

Each cluster specification in this document defines an independent functional entity. Each cluster specification is agnostic regarding functions beyond its purpose and scope, including overall requirements of the application or device. An application cluster SHOULD have no dependencies outside its application domain. A utility cluster MAY provide an interface to other layers (e.g. Groups cluster for group addressing).

Please see [Z5] Application Architecture for more details.

2.2.1.1 Cluster Identifier

A cluster identifier SHALL map to a single cluster specification. A cluster identifier also defines the purpose of a cluster instance. More than one cluster identifier, each with a unique purpose, MAY map to a single more abstract cluster specification. For example: A Concentration Measurement cluster specification MAY be quite abstract, but have many mapped cluster identifiers each with a more concrete purpose, such as CO_2 measurement in air, by volume.

Please see [Z5] Application Architecture for more details.

2.2.1.2 Extensibility Model

A cluster specification MAY be derived from a base cluster specification. A derived cluster specification SHALL add specific requirements (attributes, commands, behavior, dependencies, etc) to the base specification. A derived specification MAY reduce optionality by limiting the optional requirements from the base specification.

All new attribute and command definitions for the derived cluster SHALL be specified in the base cluster specification as optional, to maintain, in one specification, the identifier name space and communication behavior. Other behavior and dependencies that are specific to the derived cluster MAY also be specified in the base cluster, if it is deemed reusable by future derived clusters.

A derived cluster specification SHALL have the same mandatory requirements as the base cluster specification. A derived specification MAY have mandatory requirements that are optional in the base specification.

A derived cluster specification defines its own revision (*ClusterRevision* attribute) that is independent of the base specification.

Conversely, a base cluster may be defined from an original more specific cluster, which then becomes a derived cluster.

When considering the addition of one or more clusters to this specification, one SHALL explore the possibility of either deriving a cluster from an existing cluster, or creating a base cluster to map or derive new and existing cluster identifiers. This allows the reuse of approved and validated specifications and test plans.

Please see [Z5] Application Architecture for more details.

2.2.1.3 Instance Model

If a device endpoint supports both a derived server cluster identifier and its base server cluster identifier, then both SHALL represent a single instance and operate as a single entity. This makes it possible to deploy a new device endpoint with both a base and derived cluster identifiers, which SHALL remain backward compatible to legacy devices that support only the original cluster identifier.

Cluster identifiers that are mapped to a single base cluster specification, but are defined for distinctly different purposes, MAY exist together on a device endpoint. If there is no base cluster identifier defined, or no base cluster identifier exists on the same endpoint, then each cluster identifier SHALL represent a separate instance.

Please see [Z5] Application Architecture for more details.

2.2.1.4 Conformance Model

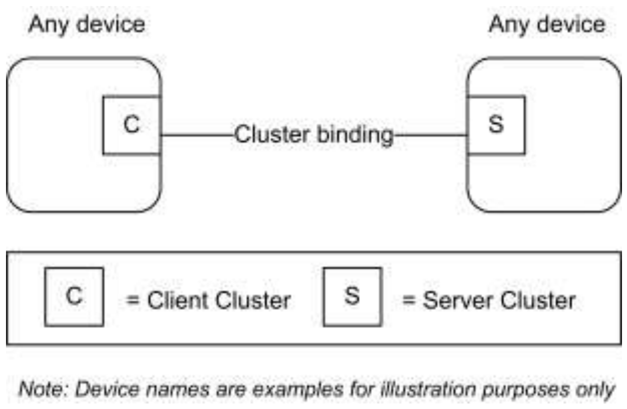
Specified behavior SHALL be Mandatory (M), Optional (O), or Deprecated (D). Mandatory behavior is usually dependent on other factors. For example: The mandatory behavior defined in a cluster server specification, is only mandatory, if the cluster id is discoverable as a server on the device. Attributes and commands MAY also be dependent on the support of other optional attributes. This is true when a feature of a cluster requires a complete set of attributes and commands.

Deprecated attributes and commands SHALL be noted as deprecated and description text SHALL be deleted.

2.2.2 Client/Server Model

Throughout the ZCL, a client/server model is employed. This model is illustrated in Figure 2-1.

Figure 2-1. The ZCL Client Server Model



A cluster is a related collection of commands and attributes, which together define an interface to specific functionality. Typically, the entity that stores the attributes of a cluster is referred to as the server of that cluster and an entity that affects or manipulates those attributes is referred to as the client of that cluster. However, if required, attributes MAY also be present on the client of a cluster.

Commands that allow devices to manipulate attributes, e.g., in this document the read attribute (see 2.5.1) or write attribute (see 2.5.3) commands, are (typically) sent from a client device and received by the server device. Any response to those commands, e.g., in this document the read attribute response (see 2.5.2) or the write attribute response (see 2.5.5 commands), are sent from the server device and received by the client device.

Conversely, the command that facilitates dynamic attribute reporting, i.e., the report attribute command (see 2.5.11) is (typically) sent from the server device (as typically this is where the attribute data itself is stored) and sent to the client device that has been bound to the server device.

A type 1 cluster's primary function is to initiate transactions from the client to the server. For example: An On/Off client sends commands (data) to the On/Off server. A type 2 cluster's primary function is to initiate transactions from the server to the client. For example: A Temperature Measurement server reports to the Temperature Measurement client. Please see [Z5] Application Architecture for more details.

The clusters supported by an application are identified through the simple descriptor (see [Z1]), specified on each active endpoint of a device. In the simple descriptor, the application input cluster list SHALL contain the list of server clusters supported on the device and the application output cluster list SHALL contain the list of client clusters supported on the device.

2.3 Functional Description

Global requirements for all clusters and commands are described here.

2.3.1 Transmission

ZCL frames are transmitted via the APS sub-layer by issuing the APSDE-DATA.request primitive.

All sub-fields of ZCL frames, including individual bits, that are unspecified, or specified as reserved, SHALL be set to zero for transmission. This applies to all ZCL frames, including cluster-specific frames. Similarly, all reserved or unspecified bits of attributes of data type class Bitmap SHALL be set to zero for transmission.

2.3.2 Reception

ZCL frames are received via the APS sub-layer by the reception of the APSDE-DATA.indication primitive.

On receipt of a command (including both general and cluster-specific commands) the device SHALL attempt to parse and execute the command. During the parsing process for a non-manufacturer-specific command, it SHALL ignore all reserved sub-fields of the ZCL frame, including individual reserved bits.

Note that, if any of these sub-fields are not set to zero, this MAY indicate that the format or interpretation of the frame has been updated. However, it is the responsibility of the specifier of such an updated format that it be backward compatible, i.e., any new format will result in the same functionality as before when parsed by a device that supports the previous version of the cluster. Any additional octets found appended to the frame SHALL also be ignored, as these MAY be added as part of such an updated frame format.

If the command is manufacturer-specific, handling of reserved sub-fields is determined by the manufacturer.

If required, the device SHALL then generate a response to the command. Responses are detailed in the specification of each command. If there is no response specified for a particular set of circumstances, (e.g., if the command has been rejected or is not recognized, or the command has succeeded but there is no response specified to indicate success), the Default Response command SHALL be generated, taking into account the conditions in 2.5.12.2. The status code returned by the Default Response command SHALL be one of the status enumerations listed in Table 2-11.

2.3.2.1 Broadcast Endpoint

The device processing a message sent to the broadcast endpoint (0xff) SHALL:

1. only deliver a copy of the message to the endpoints supporting the cluster indicated in the APS Header.
2. follow the Default Response command behavior described in section 2.5.12.2 (no response for non-unicast messages).
3. not generate error response messages, except when required by the Default Response command behavior.

2.3.2.2 Broadcast Endpoint Recommendations

Broadcast Endpoint Behavior Recommendations for Avoiding Network Congestion:

1. A device SHOULD NOT send a broadcast message to the broadcast endpoint where a response is expected from every active endpoint. It is recommended to use discovery to determine the specific endpoint(s) per device and then send individual messages that target those specific endpoints.

2. A device processing a message sent to the broadcast endpoint SHOULD jitter messages that are sent in response, especially when the nature of the message is such that it generates many responses (i.e. synchronization message).

NOTE: Multicast group messages do not include an endpoint

2.3.3 Manufacturer Specific Extensions

Manufacturers are free to extend the standard in the following ways:

- Add manufacturer specific clusters to a standard device endpoint.
- Add manufacturer specific commands to a standard cluster.
- Add manufacturer specific attributes to a standard cluster.

All communications regarding manufacturer specific extensions SHALL be transmitted with the manufacturer specific sub-field of the frame control field set to 1 and the manufacturer code included in the frame.

If the manufacturer code in a command frame is not recognized, the command is not carried out.

2.3.4 Attributes

2.3.4.1 Variables and Attributes1

A cluster variable (or variable) is a cluster data point with a defined value that is referenced in a cluster specification. A cluster attribute is a variable with a defined identifier, data type and access type. Optional attributes MAY be referenced as variables in other attribute specifications within the same cluster specification.

2.3.4.2 Dependencies on Optional Attributes

If the specification text of a cluster depends on the value of an optional attribute or variable of the same cluster, then the optional attribute or variable SHALL have a well-defined measured or default value. When an optional attribute is not supported by a device, then the default value SHALL be used for the specification text that depends on it[2]. This rule SHALL be recursive if there is a chain of dependencies.

2.3.4.3 Default Value3

If the Default Value of an attribute or variable is specified as "MS" or "ms", then there is no default value and the application must return a manufacturer specific valid value that is in the operational range. An attribute or variable SHALL have a defined default value when:

- an initial value is needed before the application starts
- the value cannot be determined by the application for the instance
- the attribute is not implemented, but there is a dependency on the attribute value

If the default value is not specified in this specification, then the default value is the Illegal Value as specified in the data type specification (see 2.6.2), if defined. If no Illegal Value is specified, then the default value SHALL be zero.

2.3.4.4 Attribute Access

Attributes MAY support these types of access that are listed in each cluster specification's attribute table for each attribute:

Access	Abrev	Description
Read	R	global commands that read the attribute value
Write	W	global commands that write a new value to the attribute
Read/Write	RW	Supports Read and Write access
Read*Write	R*W	Supports Read access. Write access as determined by the attribute implementation. If not writable, a returned status field SHALL be READ_ONLY, unless specified otherwise.
Report	P	global commands that report the value attribute or configure the attribute for reporting

[1] CCB 2327 optional attributes as dependent variables in other attribute (see Dehumidification Control)
[2] CCB 2327 generalize dependencies in specification text
[3] CCB 2370 define what a default value is

Scene	S	if a Scenes server cluster instance is on the same endpoint, then the attribute is accessed through a scene as an extension field in the scenes table

Local specific cluster commands for the cluster supporting the attribute MAY also access the attributes as defined in each cluster specification.

2.3.4.5 Global Attributes

Cluster global attributes (see 2.6.1.3) are either mandatory or optional. All cluster instances SHALL support mandatory global attributes.

Table 2-1. Global Attributes

Id	Name	Type	Range	Access	Def	M/O
0xfffd	ClusterRevision	uint16	0x0001 - 0xfffe	R	1	M
0xfffe	AttributeReportingStatus	enum8	0x00 - 0xff	R	-	O

2.3.4.5.1 ClusterRevision Attribute

The *ClusterRevision* global attribute is mandatory for all cluster instances, client and server, conforming to ZCL revision 6 (ZCL6) and later ZCL revisions. This cluster attribute represents the revision of the cluster specification that has been implemented. An implementation of a cluster specification before ZCL6 SHALL have an assumed cluster revision of 0. The initial value for the *ClusterRevision* attribute SHALL be 1. The *ClusterRevision* attribute SHALL be incremented and associated with each approved revision and release of a cluster specification.

An implementation of a new revision of a cluster specification SHALL interoperate with an implementation of an older revision of the cluster specification.

Interoperability with a cluster MAY require reading the *ClusterRevision* attribute. For example: If a new product application supporting revision 3 of cluster *X* wishes to take advantage of the new behavior that is mandated by revision 3, then the application SHOULD read the revision of the corresponding cluster *X* in each remote application. If a corresponding cluster *X* supports revision 3 or greater, than the behavior is supported. Conversely: Backward compatibility MAY require that a new cluster revision read the *ClusterRevision* of a corresponding cluster to support interoperability with legacy cluster revisions.

Please see [Z5] Application Architecture for more details.

2.3.4.5.2 AttributeReportingStatus Attribute

When reporting requires sending multiple *Report Attributes* commands, this attribute SHOULD be included in the last attribute record, to indicate that all required attributes have been reported, or that there are still attributes pending to be reported. The enumerated values for this attribute are outlined below:

Table 2-2. *AttributeReportingStatus* Enumerations

Enumerated Value	Status
0x00	Pending

0x01	Attribute Reporting Complete

2.3.5 Persistent Data

Persistent data is persistent across a restart. A restart is a program restart (warm start) or power cycle (cold start), but not a factory reset.

Cluster attributes that represent configuration data SHALL be persistent data unless otherwise specified. For example: a writeable attribute that persistently changes the behavior (or mode) of the cluster. Examples of non-configuration data: data that is calculated or comes from an external source, such as a sensor value, a time value, etc.

Many clusters define persistent data that are not attributes. For example: The scene table that is part of a Scene cluster instance, or the Alarm Table in the Alarms cluster.

Commissioning or configuration data that is created to allow the cluster to perform its function is persistent data. For example: A reporting configuration for a cluster attribute.

An APS group table entry and an APS binding are both persistent data across a restart.

A factory reset is a deliberate behavior to reset the above described persistent data back to its original state when the product left the factory.

2.4 Command Frame Formats

All commands, defined in this specification, SHALL be transmitted to the stack using the message service.

The transmission order for octets and bits of all ZCL elements is as specified in section 1.2.1.3 of the Zigbee Specification [Z1], i.e., least significant octet and bit first.

2.4.1 General ZCL Frame Format

The ZCL frame format is composed of a ZCL header and a ZCL payload. The general ZCL frame SHALL be formatted as illustrated in Figure 2-2.

Figure 2-2. Format of the General ZCL Frame

Bits: 8	**0/16**	**8**	**8**	**Variable**
Frame control	Manufacturer code	Transaction sequence number	Command identifier	Frame payload
ZCL header				ZCL payload

2.4.1.1 Frame Control Field

The frame control field is 8 bits in length and contains information defining the command type and other control flags. The frame control field SHALL be formatted as shown in Figure 2-3. Bits 5-7 are reserved for future use and SHALL be set to 0.

Figure 2-3. Format of the Frame Control Field

Bits: 0-1	**2**	**3**	**4**	**5-7**
Frame type	Manufacturer specific	Direction	Disable Default Response	Reserved

2.4.1.1.1 Frame Type Sub-field

The frame type sub-field is 2 bits in length and SHALL be set to one of the non-reserved values listed in Figure 2-4.

Figure 2-4. Values of the Frame Type Sub-field

Frame Type Value b1b0	Description
00	Command is global for all clusters, including manufacturer specific clusters
01	Command is specific or local to a cluster
other values	Reserved

2.4.1.1.2 Manufacturer Specific Sub-field

The manufacturer specific sub-field is 1 bit in length and specifies whether this command refers to a manufacturer specific extension. If this value is set to 1, the manufacturer code field SHALL be present in the ZCL frame. If this value is set to 0, the manufacturer code field SHALL not be included in the ZCL frame. Manufacturer specific clusters SHALL support global commands (Frame Type 0b00).

2.4.1.1.3 Direction Sub-field

The direction sub-field specifies the client/server direction for this command. If this value is set to 1, the command is being sent from the server side of a cluster to the client side of a cluster. If this value is set to 0, the command is being sent from the client side of a cluster to the server side of a cluster.

2.4.1.1.4 Disable Default Response Sub-field

The disable Default Response sub-field is 1 bit in length. If it is set to 0, the Default Response command will be returned, under the conditions specified in 2.5.12.2. If it is set to 1, the Default Response command will only be returned if there is an error, also under the conditions specified in 2.5.12.2.

This field SHALL be set to 1, for all response frames generated as the immediate and direct effect of a previously received frame.

2.4.1.2 Manufacturer Code Field

The manufacturer code field is 16 bits in length and specifies the assigned manufacturer code for proprietary extensions. This field SHALL only be included in the ZCL frame if the manufacturer specific sub-field of the frame control field is set to 1. Please see [Z12] Manufacturer Code Database for a list of manufacturer codes.

2.4.1.3 Transaction Sequence Number

The Transaction Sequence Number field is 8 bits in length and specifies an identification number for a single transaction that includes one or more frames in both directions. Each time the first frame of a transaction is generated, a new value SHALL be copied into the field. When a frame is generated as the specified effect on receipt of a previous frame, then it is part of a transaction, and the Transaction Sequence Number SHALL be copied from the previously received frame into the generated frame. This includes a frame that is generated in response to request frame.

The Transaction Sequence Number field can be used by a controlling device, which MAY have issued multiple commands, so that it can match the incoming responses to the relevant command.

2.4.1.4 Command Identifier Field

The Command Identifier field is 8 bits in length and specifies the cluster command being used. If the frame type sub-field of the frame control field is set to 0b00, the command identifier corresponds to one of the non-reserved values of Table 2-3. If the frame type sub-field of the frame control field is set to 0b01, the command identifier corresponds to a cluster specific command. The cluster specific command identifiers can be found in each individual document describing the clusters (see also 2.2.1.1).

2.4.1.5 Frame Payload Field

The frame payload field has a variable length and contains information specific to individual command types. The maximum payload length for a given command is limited by the stack profile in use, in conjunction with the applicable cluster specification and application profile. Fragmentation will be used where available.

2.5 General Command Frames

General command frames are used for manipulating attributes and other general tasks that are not specific to an individual cluster.

The command frames defined in this document are listed in Table 2-3. Each command frame SHALL be constructed with the frame type sub-field of the frame control field set to 0b00.

All clusters (server and client) SHALL support generation, reception and execution of the Default Response command.

Except for the optional Discover Attributes Extended commands and the optional Discover Commands commands, each cluster (server or client) that implements attributes SHALL support reception of, execution of, and response to all commands to discover, read, and write these attributes. However, if no attributes with structured types are supported, it is not required to support the structured read and write commands.

Implementation of commands to report, Configure Reporting of, and Read Reporting Configuration of attributes is only mandatory if the cluster has attributes whose reportability is mandatory.

Generation of request commands (e.g., Read Attributes, Write Attributes, etc), is application dependent.

Table 2-3. ZCL Command Frames

Command Identifier Field Value	Description
0x00	Read Attributes
0x01	Read Attributes Response
0x02	Write Attributes
0x03	Write Attributes Undivided
0x04	Write Attributes Response
0x05	Write Attributes No Response
0x06	Configure Reporting
0x07	Configure Reporting Response
0x08	Read Reporting Configuration
0x09	Read Reporting Configuration Response

0x0a	Report attributes
0x0b	Default Response
0x0c	Discover Attributes
0x0d	Discover Attributes Response
0x0e	Read Attributes Structured
0x0f	Write Attributes Structured
0x10	Write Attributes Structured response
0x11	Discover Commands Received
0x12	Discover Commands Received Response
0x13	Discover Commands Generated
0x14	Discover Commands Generated Response
0x15	Discover Attributes Extended
0x16	Discover Attributes Extended Response

2.5.1 Read Attributes Command

2.5.1.1 Read Attributes Command Frame Format

The Read Attributes command frame SHALL be formatted as illustrated in Figure 2-5.

Figure 2-5. Format of the Read Attributes Command Frame

Octets: Variable	**2**	**2**	**…**	**2**
ZCL header	Attribute identifier 1	Attribute identifier 2	…	Attribute identifier *n*

2.5.1.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Read Attributes defined for any cluster in the ZCL or 1 if this command is being used to read manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Read Attributes command (see Table 2-3).

2.5.1.1.2 Attribute Identifier Field

The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that is to be read.

2.5.1.2 When Generated

The Read Attributes command is generated when a device wishes to determine the values of one or more attributes located on another device. Each attribute identifier field SHALL contain the identifier of the attribute to be read.

2.5.1.3 Effect on Receipt

On receipt of this command, the device SHALL process each specified attribute identifier and generate a Read Attributes Response command. The Read Attributes Response command SHALL contain as many read attribute status records as attribute identifiers included in this command frame, subject to applicable space limitations. Each read attribute status record SHALL contain the corresponding attribute identifier from this command frame, a status value evaluated as described below, and, depending on the status value, the value of the attribute itself.

For each attribute identifier included in the command frame, the device SHALL create an attribute status record as follows:

If the attribute identifier does not correspond to an attribute that exists on this device, the device SHALL set the status field of the corresponding read attribute status record to UNSUPPORTED_ATTRIBUTE and SHALL not include an attribute value field.

If the attribute identified by the attribute identifier is supported, the device SHALL determine if the attribute status record carrying the attribute's current value fits into the remaining space available in the response frame. If the status record does not fit, the device SHALL set the status field of the corresponding read attribute status record to INSUFFICIENT_SPACE and not include the data type and value fields. Otherwise the device SHALL set the status field of the corresponding read attribute status record to SUCCESS and SHALL set the attribute value field to its current value.

If the resulting attribute status record does not fit into the response frame, the device SHALL transmit the response frame as assembled so far and terminate this process.

Otherwise, the device SHALL then move on to the next attribute identifier.

2.5.2 Read Attributes Response Command

2.5.2.1 Read Attributes Response Command Frame Format

The Read Attributes Response command frame SHALL be formatted as illustrated in Figure 2-6.

Figure 2-6. Format of Read Attributes Response Command Frame

Octets: Variable	Variable	Variable	…	Variable
ZCL header	Read attribute status record 1	Read attribute status record 2	…	Read attribute status record *n*

Each read attribute status record SHALL be formatted as illustrated in Figure 2-7.

Figure 2-7. Format of the Read Attributes Status Record Field

Octets: 2	1	0 / 1	0 / Variable
Attribute identifier	Status	Attribute data type	Attribute value

2.5.2.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used as a response to reading attributes defined for any cluster in the ZCL or 1 if this command is being used as a response to reading manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Read Attributes Response command (see Table 2-3).

2.5.2.1.2 Attribute Identifier Field

The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that has been read (or of which an element has been read). This field SHALL contain the same value that was included in the corresponding attribute identifier field of the original Read Attributes or Read Attributes Structured command.

2.5.2.1.3 Status Field

The status field is 8 bits in length and specifies the status of the read operation on this attribute. This field SHALL be set to SUCCESS, if the operation was successful, or an error code, as specified in 2.5.1.3, if the operation was not successful.

2.5.2.1.4 Attribute Data Type Field

The attribute data type field SHALL contain the data type of the attribute in the same Read Attributes status record (see 2.6.2).

). This field SHALL only be included if the associated status field contains a value of SUCCESS.

2.5.2.1.5 Attribute Value Field

The attribute value field is variable in length and SHALL contain the current value of this attribute. This field SHALL only be included if the associated status field contains a value of SUCCESS.

For an attribute or element of simple type (not array, structure, set or bag), this field has the format shown in the Table of Data Types (see 2.6.2). For an attribute or element of type array, set or bag, this field has the format shown in Figure 2-8.

Figure 2-8. Format of the Attribute Value Field for an Array, Set or Bag

Octets: 1	**2**	**Variable**	**…**	**Variable**
Element type	Number of elements (*m*)	Element value 1	…	Element value *m*

(NB The reason that the Element type field is before the Number of elements field is so that the latter field is in the logical position for the zeroth element.)

If the Number of elements field has the value 0xffff, this indicates that the attribute or element being read is invalid / undefined. In this case, or if the Number of elements field has the value 0, no Element value fields are included.

For an attribute or element of type structure, this field has the format shown in Figure 2-9.

Figure 2-9. Format of the Attribute Value Field for a Structure

Octets: 2	**1**	**Variable**	**…**	**1**	**Variable**
Number of elements (*m*)	Element type 1	Element value 1	…	Element type *m*	Element value *m*

In both figures, the Element value subfield follows the same format as that of the attribute value field. This format is thus recursive to any required depth (see Selector Field for limitations).

If the Number of elements field has the value 0xffff, this indicates that the attribute or element being read is invalid / undefined. In this case, or if the Number of elements field has the value 0, no Element type or Element value fields are included.

2.5.2.2 When Generated

The Read Attributes Response command is generated in response to a Read Attributes or Read Attributes Structured command. The command frame SHALL contain a read attribute status record for each attribute identifier specified in the original Read Attributes or Read Attributes Structured command. For each read attribute status record, the attribute identifier field SHALL contain the identifier specified in the original Read Attributes or Read Attributes Structured command. The status field SHALL contain a suitable status code, as detailed in 2.5.1.3.

The attribute data type and attribute value field SHALL only be included in the read attribute status record if the associated status field contains a value of SUCCESS and, where present, SHALL contain the data type and current value, respectively, of the attribute, or element thereof, that was read.

The length of this command may exceed a single frame, and thus fragmentation support may be needed to return the entire response If fragmentation is not supported, only as many read attribute status records as will fit in the frame SHALL be returned.

2.5.2.3 Effect on Receipt

On receipt of this command, the originator is notified of the results of its original Read Attributes attempt and, for each successful request, the value of the requested attribute.

If fragmentation is not supported, and some trailing attribute status records have not been returned, due to space limitations in the frame, the originator may issue an additional Read Attributes or Read Attributes Structured command to obtain their values.

2.5.3 Write Attributes Command

2.5.3.1 Write Attributes Command Frame Format

The Write Attributes command frame SHALL be formatted as illustrated in Figure 2-10.

Figure 2-10. Format of the Write Attributes Command Frame

Octets: Variable	Variable	Variable	…	Variable
ZCL header	Write attribute record 1	Write attribute record 2	…	Write attribute record *n*

Each write attribute record SHALL be formatted as illustrated in Figure 2-11.

Figure 2-11. Format of the Write Attribute Record Field

Octets: 2	1	Variable
Attribute identifier	Attribute data type	Attribute data

2.5.3.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Write Attributes defined for any cluster in the ZCL or 1 if this command is being used to write manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Write Attributes command (see Table 2-3).

2.5.3.1.2 Attribute Identifier Field

The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that is to be written.

2.5.3.1.3 Attribute Data Type Field

The attribute data type field SHALL contain the data type of the attribute that is to be written.

2.5.3.1.4 Attribute Data Field

The attribute data field is variable in length and SHALL contain the actual value of the attribute that is to be written.

2.5.3.2 When Generated

The Write Attributes command is generated when a device wishes to change the values of one or more attributes located on another device. Each write attribute record SHALL contain the identifier and the actual value of the attribute to be written.

2.5.3.3 Effect on Receipt

On receipt of this command, the device SHALL attempt to process each specified write attribute record and SHALL construct a write attribute response command (2.5.5). Each write attribute status record of the constructed command SHALL contain the identifier from the corresponding write attribute record and a status value evaluated as described below.

For each write attribute record included in the command frame, the device SHALL make the error checks listed below, in the order shown. If an error is detected, a corresponding write attribute status record SHALL be generated, the status SHALL be set according to the check below, and the device SHALL move on to the next write attribute record.

1. If the attribute is not supported on this device, the status field of the corresponding write attribute status record SHALL be set to UNSUPPORTED_ATTRIBUTE.
2. If the attribute data type field is incorrect, the device SHALL set the status field of the corresponding write attribute status record to INVALID_DATA_TYPE.
3. If the attribute is designated as read only, the device SHALL set the status field of the corresponding write attribute status record to READ_ONLY.
4. If the device is not currently accepting write attribute commands for the attribute, the status field of the corresponding write attribute status record SHALL be set to NOT_AUTHORIZED or READ_ONLY.
5. If the supplied value is not within the specified range of the attribute, the status field of the corresponding write attribute status record SHALL be set to INVALID_VALUE.
6. If the device cannot support the supplied value, the status field of the corresponding write attribute status record SHALL be set to INVALID_VALUE.

If the above error checks pass without generating a write attribute status record, the device SHALL write the supplied value to the identified attribute, and SHALL move on to the next write attribute record.

When all write attribute records have been processed, the device SHALL generate the constructed Write Attributes Response command. If there are no write attribute status records in the constructed command, indicating that all attributes were written successfully, a single write attribute status record SHALL be included in the command, with the status field set to SUCCESS and the attribute identifier field omitted.

2.5.4 Write Attributes Undivided Command

The Write Attributes Undivided command is generated when a device wishes to change the values of one or more attributes located on another device, in such a way that if any attribute cannot be written (e.g., if an attribute is not implemented on the device, or a value to be written is outside its valid range), no attribute values are changed.

In all other respects, including generation of a Write Attributes Response command, the format and operation of the command is the same as that of the Write Attributes command, except that the command identifier field SHALL be set to indicate the Write Attributes Undivided command (see Table 2-3).

2.5.5 Write Attributes Response Command

2.5.5.1 Write Attributes Response Command Frame Format

The Write Attributes Response command frame SHALL be formatted as illustrated in Figure 2-12.

Figure 2-12. Format of Write Attributes Response Command Frame

Octets: Variable	**3**	**3**	**…**	**3**
ZCL header	Write attribute status record 1	Write attribute status record 2	…	Write attribute status record *n*

Each write attribute status record SHALL be formatted as illustrated in Figure 2-13.

Figure 2-13. Format of the Write Attribute Status Record Field

Octets: 1	**2**
Status	Attribute identifier

2.5.5.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used as a response to writing attributes defined for any cluster in the ZCL or 1 if this command is being used as a response to writing manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Write Attributes Response command (see Table 2-3).

2.5.5.1.2 Status Field

The status field is 8 bits in length and specifies the status of the write operation attempted on this attribute, as detailed in 2.5.3.3.

Note that write attribute status records are not included for successfully written attributes, to save bandwidth. In the case of successful writing of all attributes, only a single write attribute status record SHALL be included in the command, with the status field set to SUCCESS and the attribute identifier field omitted.

2.5.5.1.3 Attribute Identifier Field

The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute on which the write operation was attempted.

2.5.5.2 When Generated

The Write Attributes Response command is generated in response to a Write Attributes command.

2.5.5.3 Effect on Receipt

On receipt of this command, the device is notified of the results of its original Write Attributes command.

2.5.6 Write Attributes No Response Command

2.5.6.1 Write Attributes No Response Command Frame Format

The Write Attributes No Response command frame SHALL be formatted as illustrated in Figure 2-14.

Figure 2-14. Write Attributes No Response Command Frame

Octets: Variable	**Variable**	**Variable**	**…**	**Variable**
ZCL header	Write attribute record 1	Write attribute record 2	…	Write attribute record *n*

Each write attribute record SHALL be formatted as illustrated in Figure 2-11.

2.5.6.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Write Attributes defined for any cluster in the ZCL or 1 if this command is being used to write manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Write Attributes No Response command (see Table 2-3).

2.5.6.1.2 Write Attribute Records

Each write attribute record SHALL be formatted as illustrated in Figure 2-11. Its fields have the same meaning and contents as the corresponding fields of the Write Attributes command.

2.5.6.2 When Generated

The Write Attributes No Response command is generated when a device wishes to change the value of one or more attributes located on another device but does not require a response. Each write attribute record SHALL contain the identifier and the actual value of the attribute to be written.

2.5.6.3 Effect on Receipt

There SHALL NOT be any response, error response, or Default Response command, to this command.

On receipt of this command, the device SHALL attempt to process each specified write attribute record.

For each write attribute record included in the command frame, the device SHALL first check that it corresponds to an attribute that is implemented on this device. If it does not, the device SHALL ignore the attribute and move on to the next write attribute record.

If the attribute identified by the attribute identifier is supported, the device SHALL check whether the attribute is writable. If the attribute is designated as read only, the device SHALL ignore the attribute and move on to the next write attribute record.

If the attribute is writable, the device SHALL check that the supplied value in the attribute data field is within the specified range of the attribute. If the supplied value does not fall within the specified range of the attribute, the device SHALL ignore the attribute and move on to the next write attribute record.

If the value supplied in the attribute data field is within the specified range of the attribute, the device SHALL write the supplied value to the identified attribute and move on to the next write attribute record.

2.5.7 Configure Reporting Command

The Configure Reporting command is used to configure the reporting mechanism for one or more of the attributes of a cluster.

The individual cluster definitions specify which attributes SHALL be available to this reporting mechanism, however specific implementations of a cluster may make additional attributes available.

Note that attributes with data types of array, structure, set or bag cannot be reported.

2.5.7.1 Configure Reporting Command Frame Format

The Configure Reporting command frame SHALL be formatted as illustrated in Figure 2-15.

Figure 2-15. Format of the Configure Reporting Command Frame

Octets: Variable	Variable	Variable	…	Variable
ZCL header	Attribute reporting configuration record 1	Attribute reporting configuration record 2	…	Attribute reporting configuration record *n*

There SHALL be one attribute reporting configuration record for each attribute to be configured. Each such record SHALL be formatted as illustrated in Figure 2-16.

Figure 2-16. Format of the Attribute Reporting Configuration Record

Octets: 1	2	0/1	0/2	0/2	0/Variable	0/2
Direction	Attribute identifier	Attribute data type	Minimum reporting interval	Maximum reporting interval	Reportable change	Timeout period

2.5.7.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to configure attribute reports defined for any cluster in the ZCL or 1 if this command is being used to configure attribute reports for manufacturer specific attributes.

The command identifier field SHALL be set to indicate the report configuration command (see Table 2-3).

2.5.7.1.2 Direction Field

The direction field specifies whether values of the attribute are to be reported, or whether reports of the attribute are to be received.

If this value is set to 0x00, then the attribute data type field, the minimum reporting interval field, the maximum reporting interval field and the reportable change field are included in the payload, and the timeout period field is omitted. The record is sent to a cluster server (or client) to configure how it sends reports to a client (or server) of the same cluster.

If this value is set to 0x01, then the timeout period field is included in the payload, and the attribute data type field, the minimum reporting interval field, the maximum reporting interval field and the reportable change field are omitted. The record is sent to a cluster client (or server) to configure how it SHOULD expect reports from a server (or client) of the same cluster.

All other values of this field are reserved.

Table 2-4. Destination of Reporting Based on Direction Field

Direction Field	Destinations
0x00	The receiver of the Configure Reporting command SHALL Configure Reporting to send to each destination as resolved by the bindings for the cluster hosting the attributes to be reported.
0x01	This indicates to the receiver of the Configure Reporting command that the sender has configured its reporting mechanism to transmit reports and that, based on the current state of the sender’s bindings, the sender will send reports to the receiver.

2.5.7.1.3 Attribute Identifier Field

If the direction field is 0x00, this field contains the identifier of the attribute that is to be reported. If instead the direction field is 0x01, the device SHALL expect reports of values of this attribute.

2.5.7.1.4 Attribute Data Type Field

The Attribute data type field contains the data type of the attribute that is to be reported.

2.5.7.1.5 Minimum Reporting Interval Field

The minimum reporting interval field is 16 bits in length and SHALL contain the minimum interval, in seconds, between issuing reports of the specified attribute.

If this value is set to 0x0000, then there is no minimum limit, unless one is imposed by the specification of the cluster using this reporting mechanism or by the application.

2.5.7.1.6 Maximum Reporting Interval Field

The maximum reporting interval field is 16 bits in length and SHALL contain the maximum interval, in seconds, between issuing reports of the specified attribute.

If this value is set to 0xffff, then the device SHALL not issue reports for the specified attribute, and the configuration information for that attribute need not be maintained. (**Note:** in an implementation using dynamic memory allocation, the memory space for that information may then be reclaimed).

If this value is set to 0x0000, and the minimum reporting interval field does not equal 0xffff there SHALL be no periodic reporting, but change based reporting SHALL still be operational.[4]

If this value is set to 0x0000 and the Minimum Reporting Interval Field equals 0xffff, then the device SHALL revert to its default reporting configuration. The reportable change field, if present, SHALL be set to zero.

2.5.7.1.7 Reportable Change Field

The reportable change field SHALL contain the minimum change to the attribute that will result in a report being issued. This field is of variable length. For attributes with 'analog' data type (see 2.6.2), the field has the same data type as the attribute. The sign (if any) of the reportable change field is ignored.

For attributes of 'discrete' data type (see 2.6.2), this field is omitted.

If the Maximum Reporting Interval Field is set to 0xffff (terminate reporting configuration)[5], or the Maximum Reporting Interval Field is set to 0x0000 and the Minimum Reporting Interval Field equals 0xffff, indicating a (default reporting configuration) then if this field is present, it SHALL be set to zero upon transmission and ignored upon reception.

2.5.7.1.8 Timeout Period Field

The timeout period field is 16 bits in length and SHALL contain the maximum expected time, in seconds, between received reports for the attribute specified in the attribute identifier field. If more time than this elapses between reports, this may be an indication that there is a problem with reporting.

If this value is set to 0x0000, reports of the attribute are not subject to timeout.

Note that, for a server/client connection to work properly using automatic reporting, the timeout value set for attribute reports to be received by the client (or server) cluster must be set somewhat higher than the maximum reporting interval set for the attribute on the server (or client) cluster.

2.5.7.2 When Generated

The report configuration command is generated when a device wishes to configure a device to automatically report the values of one or more of its attributes, or to receive such reports.

2.5.7.3 Effect on Receipt

On receipt of this command, the device SHALL attempt to process each attribute reporting configuration record and SHALL construct a Configure Reporting Response command. Each attribute status record of the constructed command SHALL contain an identifier from an attribute reporting configuration record and a status value evaluated as described below.

If the direction field is 0x00, indicating that the reporting intervals and reportable change are being configured, then

- If the attribute specified in the attribute identifier field is not implemented on this device or if the attribute type is set to array, structure, set or bag, the device SHALL construct an attribute status record with the status field set to UNSUPPORTED_ATTRIBUTE.
- Else, if the attribute identifier in this field cannot be reported (because it is not in the list of mandatory reportable attributes in the relevant cluster specification, and support has also not been implemented as a manufacturer option), the device SHALL construct an attribute status record with the status field set to UNREPORTABLE_ATTRIBUTE.

[4] Moved from Report Attributes command description
[5] CCB 2338 clarify the value of this field as dependent on the Maximum Reporting Interval field

- Else, if the attribute data type field is incorrect, the device SHALL construct an attribute status record with the status field set to INVALID_DATA_TYPE.
- Else, if the minimum reporting interval field is less than any minimum set by the relevant cluster specification or application, or the value of the maximum reporting interval field is non-zero and is less than that of the minimum reporting interval field, the device SHALL construct an attribute status record with the status field set to INVALID_VALUE.
- Else, if the value of the minimum or maximum reporting interval field is not supported by the product, the device SHALL construct an attribute status record with the status field set to INVALID_VALUE.
- Else the device SHALL set the minimum and maximum reporting intervals and the reportable change for the attribute to the values contained in the corresponding fields.

Else the direction field is 0x01, indicating that the timeout period is being configured, then

If reports of values of the attribute identifier specified in the attribute identifier field cannot be received (because it is not in the list of mandatory reportable attributes in the relevant cluster specification, and support has also not been implemented as a manufacturer option), or the timeout feature is not supported, the device SHALL construct an attribute status record with the status field set to UNSUPPORTED_ATTRIBUTE.

Else the device SHALL set the timeout value for the attribute identifier specified in the attribute identifier field to the value of the timeout period field. Note that the action to be taken by the device if the timeout period is exceeded is cluster and device dependent, including optionally taking no action.

When all attribute reporting configuration records have been processed, the device SHALL generate the constructed Configure Reporting Response command. If there are no attribute status records in the constructed command, indicating that all attributes were configured successfully, a single attribute status record SHALL be included in the command, with the status field set to SUCCESS and the direction and attribute identifier fields omitted.

The device SHALL then proceed to generate or receive attribute reports according the configuration just set up, by means of the Report Attributes command (see 2.5.11.2.1 through2.5.11.2.4). See Table 2-4 to determine the destination of the Report Attributes command.

2.5.8 Configure Reporting Response Command

The Configure Reporting Response command is used to respond to a Configure Reporting command.

2.5.8.1 Configure Reporting Response Command Frame Format

The Configure Reporting Response command frame SHALL be formatted as illustrated in Figure 2-17.

Figure 2-17. Format of the Configure Reporting Response Command Frame

Octets: Variable	4	4	…	4
ZCL header	Attribute status record 1	Attribute status record 2	…	Attribute status record *n*

Each attribute status record SHALL be formatted as illustrated in Figure 2-18.

Figure 2-18. Format of the Attribute Status Record Field

Octets: 1	1	2
Status	Direction	Attribute identifier

2.5.8.1.1 ZCL Header Fields

The frame control field is specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used as a response to configuring attribute reports defined for any cluster in the ZCL or 1 if this command is being used as a response to configuring attribute reports for manufacturer specific attributes.

The command identifier field SHALL be set to indicate the report configuration response command (see Table 2-3).

2.5.8.1.2 Direction Field

The direction field specifies whether values of the attribute are reported (0x00), or whether reports of the attribute are received (0x01).

All other values of this field are reserved.

2.5.8.1.3 Status Field

The status field specifies the status of the Configure Reporting operation attempted on this attribute, as detailed in 2.5.7.3.

Note that attribute status records are not included for successfully configured attributes, to save bandwidth. In the case of successful configuration of all attributes, only a single attribute status record SHALL be included in the command, with the status field set to SUCCESS and the direction and attribute identifier fields omitted.

2.5.8.2 When Generated

The Configure Reporting Response command is generated in response to a Configure Reporting command.

2.5.8.3 Effect on Receipt

On receipt of this command, the device is notified of the success (or otherwise) of its original Configure Reporting command, for each attribute.

2.5.9 Read Reporting Configuration Command

The Read Reporting Configuration command is used to read the configuration details of the reporting mechanism for one or more of the attributes of a cluster.

2.5.9.1 Read Reporting Configuration Command Frame Format

The Read Reporting Configuration command frame SHALL be formatted as illustrated in Figure 2-19.

Figure 2-19. Read Reporting Configuration Command Frame

Octets: Variable	**3**	**3**	**…**	**3**
ZCL header	Attribute record 1	Attribute record 2	…	Attribute record *n*

Each attribute record SHALL be formatted as illustrated in Figure 2-20.

Figure 2-20. Format of the Attribute Status Record Field

Octets: 1	2
Direction	Attribute identifier

2.5.9.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to read the reporting configuration of attributes defined for any cluster in the ZCL or 1 if this command is being used to read the reporting configuration of manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Read Reporting Configuration command (see Table 2-3).

2.5.9.1.2 Direction Field

The direction field specifies whether values of the attribute are reported (0x00), or whether reports of the attribute are received (0x01).

All other values of this field are reserved.

2.5.9.1.3 Attribute Identifier Field

The attribute identifier field SHALL contain the identifier of the attribute whose reporting configuration details are to be read.

2.5.9.2 Effect on Receipt

On receipt of this command, a device SHALL generate a Read Reporting Configuration Response command containing the details of the reporting configuration for each of the attributes specified in the command (see 2.5.10).

2.5.10 Read Reporting Configuration Response Command

The Read Reporting Configuration Response command is used to respond to a Read Reporting Configuration command.

2.5.10.1 Read Reporting Configuration Response Command Frame Format

The Read Reporting Configuration Response command frame SHALL be formatted as illustrated in Figure 2-21.

Figure 2-21. Format of the Read Reporting Configuration Response Command Frame

Octets: Variable	Variable	Variable	…	Variable
ZCL header	Attribute reporting configuration record 1	Attribute reporting configuration record 2	…	Attribute reporting configuration record *n*

There SHALL be one attribute reporting configuration record for each attribute record of the received Read Reporting Configuration command. Each such record SHALL be formatted as illustrated in Figure 2-22.

Figure 2-22. Attribute Reporting Configuration Record Field

Octets: 1	1	2	0/1	0/2	0/2	0/Variable	0/2
Status	Direction	Attribute identifier	Attribute data type	Minimum reporting interval	Maximum reporting interval	Reportable change	Timeout period

2.5.10.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to for attributes specified in the ZCL or 1 if this command is being used for manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Read Reporting Configuration Response command (see Table 2-3).

2.5.10.1.2 Status Field

If the attribute is not implemented on the sender or receiver of the command, whichever is relevant (depending on direction), this field SHALL be set to UNSUPPORTED_ATTRIBUTE. If the attribute is supported, but is not capable of being reported, this field SHALL be set to UNREPORTABLE_ATTRIBUTE. If the attribute is supported and reportable, but there is no report configuration, this field SHALL be set to NOT_FOUND. Otherwise, this field SHALL be set to SUCCESS.

If the status field is not set to SUCCESS, all fields except the direction and attribute identifier fields SHALL be omitted.

2.5.10.1.3 Direction Field

The direction field specifies whether values of the attribute are reported (0x00), or whether reports of the attribute are received (0x01).

If this value is set to 0x00, then the attribute data type field, the minimum reporting interval field, the maximum reporting interval field and the reportable change field are included in the payload, and the timeout period field is omitted. If this value is set to 0x01, then the timeout period field is included in the payload, and the attribute data type field, the minimum reporting interval field, the maximum reporting interval field and the reportable change field are omitted.

All other values of this field are reserved.

2.5.10.1.4 Attribute Identifier Field

The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that the reporting configuration details apply to.

2.5.10.1.5 Minimum Reporting Interval Field

The minimum reporting interval field is 16 bits in length and SHALL contain the minimum interval, in seconds, between issuing reports for the attribute specified in the attribute identifier field. If the minimum reporting interval has not been configured, this field SHALL contain the value 0xffff.

2.5.10.1.6 Maximum Reporting Interval Field

The maximum reporting interval field is 16 bits in length and SHALL contain the maximum interval, in seconds, between issuing reports for the attribute specified in the attribute identifier field. If the maximum reporting interval has not been configured, this field SHALL contain the value 0xffff.

2.5.10.1.7 Reportable Change Field

The reportable change field SHALL contain the minimum change to the attribute that will result in a report being issued. For attributes with 'analog' data type (see 2.6.2), the field has the same data type as the attribute. If the reportable change has not been configured, this field SHALL contain the invalid value for the relevant data type.

For attributes of 'discrete' data (see 2.6.2), this field is omitted.

2.5.10.1.8 Timeout Period Field

The timeout period field is 16 bits in length and SHALL contain the maximum expected time, in seconds, between received reports for the attribute specified in the attribute identifier field. If the timeout period has not been configured, this field SHALL contain the value 0xffff.

2.5.10.2 When Generated

The Read Reporting Configuration Response command is generated in response to a Read Reporting Configuration command. Only as many attribute reporting configuration records as will fit in the frame SHALL be returned.

2.5.10.3 Effect on Receipt

On receipt of this command, the originator is notified of the results of its original Read Reporting Configuration command.

If some trailing attribute reporting configuration records have not been returned, due to space limitations in the frame, the originator may issue a further Read Reporting Configuration command to obtain their values.

2.5.11 Report Attributes Command

The Report Attributes command is used by a device to report the values of one or more of its attributes to another device. Individual clusters, defined elsewhere in the ZCL, define which attributes are to be reported and at what interval. See 2.5.7 to determine the destination of the Report Attributes command.

2.5.11.1 Report Attributes Command Frame Format

The Report Attributes command frame SHALL be formatted as illustrated in Figure 2-23.

Figure 2-23. Format of the Report Attributes Command Frame

Octets: Variable	Variable	Variable	…	Variable
ZCL header	Attribute report 1	Attribute report 2	…	Attribute report *n*

Each attribute report field SHALL be formatted as illustrated in Figure 2-24.

Figure 2-24. Format of the Attribute Report Fields

Octets: 2	1	Variable
Attribute identifier	Attribute data type	Attribute data

2.5.11.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Report Attributes defined for any cluster in the ZCL or 1 if this command is being used to report manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Report Attributes command (see Table 2-3).

2.5.11.1.2 Attribute Identifier Field

The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that is being reported. When reporting requires sending multiple *Report Attributes* commands see 2.3.4.5.2.

2.5.11.1.3 Attribute Data Type Field

The attribute data type field contains the data type of the attribute that is being reported.

2.5.11.1.4 Attribute Data Field

The attribute data field is variable in length and SHALL contain the actual value of the attribute being reported.

2.5.11.2 When Generated

The Report Attributes command is generated when a device has been configured to report the values of one or more of its attributes to another device., and when the conditions that have been configured are satisfied. These conditions are detailed in the following sections.

A Report Attributes command may also be configured locally on a device at any time. Except for the source, a locally created report configuration SHALL be no different than a configuration received externally. A locally created report configuration SHALL support the same services as a configuration received externally.

If the destination of the Report Attributes Command cannot be determined, then the command SHALL not be generated. See 2.5.7 to determine the destination of the Report Attributes command.

2.5.11.2.1 Periodic Reporting

A report SHALL be generated when the time that has elapsed since the previous report of the same attribute is equal to the Maximum Reporting Interval for that attribute (see 2.5.7.1.6). The time of the first report after configuration is not specified.

2.5.11.2.2 Changes to 'Discrete' Attributes

If the attribute has a 'discrete' data type, a report SHALL be generated when the attribute undergoes any change of value. Discrete types are general data types (which are often used as sets of bit fields), logical types, bitmap types, enumerations, strings, identifiers, IEEE address and security key (see 2.6.2).

Reporting is subject to the Minimum Reporting Interval for that attribute (see 2.5.7.1.5). After a report, no further reports are sent during this interval.

2.5.11.2.3 Changes to 'Analog' Attributes

If the attribute has an 'analog' data type, a report SHALL be generated when the attribute undergoes a change of value, in a positive or negative direction, equal to or greater than the Reportable Change for that attribute (see 2.5.7.1.7). The change is measured from the value of the attribute when the Reportable Change is configured, and thereafter from the previously reported value of the attribute.

Analog types are signed and unsigned integer types, floating point types and time types (see 2.6.2).

Reporting is subject to the Minimum Reporting Interval for that attribute (see 2.5.7.1.5). After a report, no further reports are sent during this interval.

2.5.11.2.4 Cluster Specific Conditions

The specification for a cluster may add additional conditions for specific attributes of that cluster.

2.5.11.2.5 Consolidation of Attribute Reporting

To reduce the resources (such as the number of timers) required for attribute reporting, a device may adapt the timing of reports by relaxing the configured minimum and maximum periods as described below. By employing these techniques, a device may limit the number of timers required to any manufacturer specific value, including use of only a single timer, though at the cost of some side effects, such as increased network traffic in some cases.

In consolidating timers, several principles apply:

1. The maximum reporting interval of an attribute may be reduced, as it SHOULD not normally cause a problem to devices to receive reports more frequently than expected – typical reporting intervals are seconds to minutes. It may not be increased, as this may be incompatible with any timeout period set.
2. The minimum reporting interval of an attribute may also be reduced. However, it may not be increased, as an application may be relying on receiving reports of changes to an attribute within a given delay time. Minimum values are generally used to reduce network traffic, but this is less important than ensuring that the application timing needs are satisfied.
3. From (1), when consolidating the maximum reporting periods of two or more attributes together, the consolidated reporting period SHALL be equal to the lowest of the configured maximum intervals of the attributes to be reported.
4. Similarly, from (2), when consolidating the minimum reporting periods of two or more attributes together, the consolidated reporting period SHALL be equal to the lowest of the configured minimum intervals of the attributes to be reported.

As a first step, timers for attributes on the same cluster may be consolidated. Such adaptations SHOULD aim to send attribute reports for different attributes of the same cluster at the same time, so that they can be consolidated into fewer attribute reports, thus reducing network traffic.

To reduce the number of timers further, timers may be consolidated across clusters and endpoints if needed.

(Note that it is not generally possible to consolidate timeout values (see 2.5.7.1.8) of received attribute reports.)

2.5.11.3 Effect on Receipt

On receipt of this command, a device is notified of the latest values of one or more of the attributes of another device.

2.5.12 Default Response Command

2.5.12.1 Default Response Command Frame Format

The Default Response command frame SHALL be formatted as illustrated in Figure 2-25.

Figure 2-25. Format of the Default Response Command Frame

Octets: Variable	1	1
ZCL header	Command identifier	Status code

2.5.12.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being sent in response to a command defined for any cluster in the ZCL or 1 if this command is being sent in response to a manufacturer specific command.

The command identifier sub-field SHALL be set to indicate the Default Response command (see Table 2-3).

2.5.12.1.2 Command Identifier Field

The command identifier field is 8 bits in length and specifies the identifier of the received command to which this command is a response.

2.5.12.1.3 Status Code Field

The status code field is 8 bits in length and specifies either SUCCESS or the nature of the error that was detected in the received command. It SHALL be one of the status enumerations listed in Table 2-11.

2.5.12.2 When Generated

The Default Response command SHALL be generated when all 4 of these criteria are met:

1. A device receives a unicast command that is not a Default Response command.
2. No other command is sent in response to the received command, using the same Transaction sequence number as the received command.
3. The Disable Default Response bit of its Frame control field is set to 0 (see 2.4.1.1.4) or when an error results.
4. The “Effect on Receipt” clause for the received command does not override the behavior of when a Default Response command is sent.

If a device receives a command in error through a broadcast or multicast transmission, the command SHALL be discarded and the Default Response command SHALL not be generated.

If the identifier of the received command is not supported on the device, it SHALL set the command identifier field to the value of the identifier of the command received in error. The status code field SHALL be set to the either: UNSUP_CLUSTER_COMMAND,
UNSUP_GENERAL_COMMAND,
UNSUP_MANUF_CLUSTER_COMMAND or UNSUP_MANUF_GENERAL_COMMAND, as appropriate.

If the device receives a unicast cluster command to a particular endpoint, and the cluster does not exist on the endpoint, the status code field SHALL be set to UNSUPPORTED_CLUSTER. Receiving devices SHOULD accept other error status codes, such as FAILURE, from devices certified before ZCL revision 6.

The Default Response command SHALL be generated in response to reception of all commands, including response commands (such as the Write Attributes Response command), under the conditions specified above. However, the Default Response command SHALL not be generated in response to reception of another Default Response command.

2.5.12.3 Effect on Receipt

On receipt of this command, the device is notified of the success or otherwise of the generated command with the same transaction sequence number (see 2.4.1.3).

2.5.13 Discover Attributes Command

2.5.13.1 Discover Attributes Command Frame Format

The Discover Attributes command frame SHALL be formatted as illustrated in Figure 2-26.

Figure 2-26. Format of the Discover Attributes Command Frame

Octets: Variable	**2**	**1**
ZCL header	Start attribute identifier	Maximum attribute identifiers

2.5.13.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 to discover standard attributes in a cluster or 1 to discover manufacturer specific attributes in either a standard or a manufacturer specific cluster.

The command identifier field SHALL be set to indicate the Discover Attributes command (see Table 2-3).

2.5.13.1.2 Start Attribute Identifier Field

The start attribute identifier field is 16 bits in length and specifies the value of the identifier at which to begin the attribute discovery.

2.5.13.1.3 Maximum Attribute Identifiers Field

The maximum attribute identifiers field is 8 bits in length and specifies the maximum number of attribute identifiers that are to be returned in the resulting Discover Attributes Response command.

2.5.13.2 When Generated

The Discover Attributes command is generated when a remote device wishes to discover the identifiers and types of the attributes on a device which are supported within the cluster to which this command is directed.

2.5.13.3 Effect on Receipt

On receipt of this command, the device SHALL construct an ordered list of attribute information records, each containing a discovered attribute identifier and its data type, in ascending order of attribute identifiers. This list SHALL start with the first attribute that has an identifier that is equal to or greater than the identifier specified in the start attribu**te identifier field. The number of attribute identifiers included in the list SHALL not exceed that specified in the maximum attribute identifiers field.**

The device SHALL then generate a Discover Attributes Response command containing the discovered attributes and their types, and SHALL return it to the originator of the Discover Attributes command.

2.5.14 Discover Attributes Response Command

2.5.14.1 Discover Attributes Response Command Frame Format

The Discover Attributes Response command frame SHALL be formatted as illustrated in Figure 2-27.

Figure 2-27. Discover Attributes Response Command Frame

Octets: Variable	**1**	**3**	**3**	**…**	**3**
ZCL header	Discovery complete	Attribute information 1	Attribute information 2	…	Attribute information *n*

Each attribute information field SHALL be formatted as illustrated in Figure 2-28.

Figure 2-28. Format of the Attribute Report Fields

Octets: 2	**1**
Attribute identifier	Attribute data type

2.5.14.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to the same value included in the original Discover Attributes command.

The command identifier field SHALL be set to indicate the Discover Attributes Response command (see Table 2-3).

2.5.14.1.2 Discovery Complete Field

The discovery complete field is a Boolean field. A value of 0 indicates that there are more attributes to be discovered that have an attribute identifier value greater than the last attribute identifier in the last attribute information field. A value of 1 indicates that there are no more attributes to be discovered.

2.5.14.1.3 Attribute Identifier Field

The attribute identifier field SHALL contain the identifier of a discovered attribute. Attributes SHALL be included in ascending order, starting with the lowest attribute identifier that is greater than or equal to the start attribute identifier field of the received Discover Attributes command.

2.5.14.1.4 Attribute Data Type Field

The attribute data type field SHALL contain the data type of the attribute in the same attribute report field (see 2.6.2).

2.5.14.2 When Generated

The Discover Attributes Response command is generated in response to a Discover Attributes command.

2.5.14.3 Effect on Receipt

On receipt of this command, the device is notified of the results of its attribute discovery request.

Following the receipt of this command, if the discovery complete field indicates that there are more attributes to be discovered, the device may choose to send subsequent discover attribute request commands to obtain the rest of the attribute identifiers. In this case, the start attribute identifier specified in the next attribute discovery request command SHOULD be set equal to one plus the last attribute identifier received in the Discover Attributes Response command.

2.5.15 Read Attributes Structured Command

2.5.15.1 Read Attributes Structured Command Frame Format

The Read Attributes Structured command frame SHALL be formatted as illustrated in Figure 2-29.

Figure 2-29. Format of Read Attributes Structured Command Frame

Octets: Variable	2	Variable	...	2	Variable
ZCL header	Attribute identifier 1	Selector 1	...	Attribute identifier *n*	Selector *n*

2.5.15.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Read Attributes defined for any cluster in the ZCL or 1 if this command is being used to read manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Read Attributes Structured command (see Table 2-3).

2.5.15.1.2 Attribute Identifier Field

The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that is to be read.

2.5.15.1.3 Selector Field

Each attribute identifier field is followed by a selector field, which specifies whether the whole of the attribute value is to be read, or only an individual element of it. An individual element may only be read from attributes with types of Array or Structure.

The Selector field SHALL be formatted as illustrated in Figure 2-30.

Figure 2-30. Format of the Selector Field

Octets: 1	2	...	2
Indicator (*m*)	Index 1	...	Index *m*

The Indicator subfield indicates the number of index fields that follow it. This number is limited to the range 0 - 15. It may be further limited by an application. All other values of this field are reserved.

If this subfield is 0, there are no index fields, and the whole of the attribute value is to be read. For attributes of type other than array or structure, this subfield SHALL have the value 0.

If this subfield is 1 or greater, the index fields indicate which element is to be read, nested to a depth of m. For example, if the attribute is an array of arrays (or structures), then if m = 2, index 1 = 5 and index 2 = 3, the third element of the fifth element of the attribute will be read.

Note that elements are numbered from 1 upwards for both arrays and structures. The zeroth element of an array or structure is readable, always has type 16 bit unsigned integer, and returns the number of elements contained in the array or structure.

2.5.15.2 When Generated

The Read Attributes command is generated when a device wishes to determine the values of one or more attributes, or elements of attributes, located on another device. Each attribute identifier field SHALL contain the identifier of the attribute to be read.

2.5.15.3 Effect on Receipt

On receipt of this command, the device SHALL process each specified attribute identifier and associated selector, and SHALL generate a Read Attributes Response command. The Read Attributes Response command SHALL contain as many read attribute status records as there are attribute identifiers included in this command frame. Each read attribute status record SHALL contain the corresponding attribute identifier from this command frame, a status value evaluated as described below, and, depending on the status value, the value of the attribute (or attribute element) itself.

For each attribute identifier included in the command frame, the device SHALL first check that it corresponds to an attribute that exists on this device, and that its associated selector field correctly indicates either the whole of the attribute or an element of the attribute. If it does not, the device SHALL set the status field of the corresponding read attribute status record to either UNSUPPORTED_ATTRIBUTE or INVALID_SELECTOR as appropriate, and SHALL not include an attribute value field. The device SHALL then move on to the next attribute identifier.

If the attribute identified by the attribute identifier is supported, and its associated selector field is valid, the device SHALL set the status field of the corresponding read attribute status record to SUCCESS and SHALL set the attribute value field to the value of the attribute (or its selected element). The device SHALL then move on to the next attribute identifier.

2.5.16 Write Attributes Structured Command

2.5.16.1 Write Attributes Structured Command Frame Format

The Write Attributes Structured command frame SHALL be formatted as illustrated in Figure 2-31.

Figure 2-31. Write Attributes Structured Command Frame

Octets: Variable	Variable	Variable	...	Variable
ZCL header	Write attribute record 1	Write attribute record 2	...	Write attribute record *n*

Each write attribute record SHALL be formatted as illustrated in Figure 2-32.

Figure 2-32. Format of the Write Attribute Record Field

Octets: 2	Variable	1	Variable
Attribute identifier	Selector	Attribute data type	Attribute value

2.5.16.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Write Attributes defined for any cluster in the ZCL or 1 if this command is being used to write manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Write Attributes Structured command (see Table 2-3).

2.5.16.1.2 Attribute Identifier Field

The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that is to be written (or an element of which is to be written).

2.5.16.1.3 Selector Field

The selector field specifies whether the whole of the attribute value is to be written, or only an individual element of it. An individual element may only be written to attributes with types of Array, Structure, Set or Bag.

The Selector field SHALL be formatted as illustrated in Figure 2-33.

Figure 2-33. Format of the Selector Field

Octets: 1	2	...	2
Indicator (*m*)	Index 1	...	Index *m*

2.5.16.1.4 Writing an Element to an Array or Structure

When writing an element to an array or structure, the Indicator subfield indicates the number of index fields that follow it. This number is limited to the range 0 - 15 (i.e., the upper 4 bits of the Indicator field are set to zero). It may be further limited by an application.

If the Indicator subfield is 0, there are no index fields, and the whole of the attribute value is to be written.

If this subfield is 1 or greater, the index fields indicate which element is to be written, nested to a depth of m. For example, if the attribute is an array of arrays (or structures), then if m = 2, index 1 = 5 and index 2 = 3, the third element of the fifth element of the attribute will be written.

Note that elements are numbered from 1 upwards for both arrays and structures.

The zeroth element of an array or structure has type 16-bit unsigned integer, and holds the number of elements in the array or structure. The zeroth element of an array may optionally be written (this is application dependent) and has the effect of changing the number of elements of the array. If the number is reduced, the array is truncated. If the number is increased, the content of new elements is application dependent.

The zeroth element of a structure may not be written to. Writing to an element with an index greater than the number of elements in an array or structure is always an error.

2.5.16.1.5 Adding/Removing an Element to/from a Set or Bag

This command may also be used to add an element to a set or bag, or to remove an element from a set or bag.

In this case, the lower 4 bits of the Indicator subfield still indicate the number of index fields that follow it, as the set may be an element of an array or structure, which may itself be nested inside other arrays or structures.

The upper 4 bits of the Indicator subfield have the following values:

0b0000	Write whole set/bag
0b0001	Add element to the set/bag
0b0010	Remove element from the set/bag

All other values are reserved.

2.5.16.1.6 Attribute Data Type Field

The attribute data type field SHALL contain the data type of the attribute or element thereof that is to be written.

2.5.16.1.7 Attribute Value Field

The attribute value field is variable in length and SHALL contain the actual value of the attribute, or element thereof, that is to be written. For an attribute or element of type array, structure, set or bag, this field has the same format as for the Read Attributes Structured command (see Read Attributes Structured Command).

2.5.16.2 When Generated

The Write Attributes Structured command is generated when a device wishes to change the values of one or more attributes located on another device. Each write attribute record SHALL contain the identifier and the actual value of the attribute, or element thereof, to be written.

2.5.16.3 Effect on Receipt

On receipt of this command, the device SHALL attempt to process each specified write attribute record and SHALL construct a write attribute structured response command. Each write attribute status record of the constructed command SHALL contain the identifier from the corresponding write attribute record and a status value evaluated as described below.

For each write attribute record included in the command frame, the device SHALL first check that it corresponds to an attribute that is implemented on this device and that its associated selector field correctly indicates either the whole of the attribute or an element of the attribute. If it does not (e.g., an index is greater than the number of elements of an array), the device SHALL set the status field of the corresponding write attribute status record to either UNSUPPORTED_ATTRIBUTE or INVALID_SELECTOR as appropriate and move on to the next write attribute record.

If the attribute identified by the attribute identifier is supported, the device SHALL check whether the attribute data type field is correct. (**Note:** If the element being written is the zeroth element of an array (to change the length of the array) the data type must be 16-bit unsigned integer). If not, the device SHALL set the status field of the corresponding write attribute status record to INVALID_DATA_TYPE and move on to the next write attribute record.

If the attribute data type is correct, the device SHALL check whether the attribute is writable. If the attribute is designated as read only, the device SHALL set the status field of the corresponding write attribute status record to READ_ONLY and move on to the next write attribute record. (**Note:** If an array may not have its length changed, its zeroth element is read only).

If the attribute is writable, the device SHALL check that all the supplied basic (e.g., integer, floating point) values in the attribute value field are within the specified ranges of the elements they are to be written to. If a supplied value does not fall within the specified range of its target element, the device SHALL set the status field of the corresponding write attribute status record to INVALID_VALUE, SHALL set the selector field of that record to indicate that target element, and SHALL move on to the next write attribute record.

The returned selector SHALL have the number of indices necessary to specify the specific low-level element that failed, which will be the same as or greater than the number of indices in the selector of the write attribute record. Note that if the element being written is the zeroth element of an array (to change the length of the array) and the requested new length is not acceptable to the application, the value being written is considered outside the specified range of the element.

If the value supplied in the attribute value field is within the specified range of the attribute, the device SHALL proceed as follows.

- If an element is being added to a set, and there is an element of the set that has the same value as the value to be added, the device SHALL set the status field of the corresponding write attribute status record to DUPLICATE_ENTRY and move on to the next write attribute record.
- Else, if an element is being removed from a set or a bag, and there is no element of the set or bag that has the same value as the value to be removed, the device SHALL set the status field of the corresponding write attribute status record to NOT_FOUND and move on to the next write attribute record.
- Otherwise, the device SHALL write, add or remove the supplied value to/from the identified attribute or element, as appropriate, and SHALL move on to the next write attribute record. In this (successful) case, a write attribute status record SHALL not be generated. (**Note:** If the element being written is the zeroth element of an array, the length of the array SHALL be changed. If the length is reduced, the array is truncated. If the length is increased, the content of new elements is application dependent.)

When all write attribute records have been processed, the device SHALL generate the constructed Write Attributes Response command. If there are no write attribute status records in the constructed command, because all attributes were written successfully, a single write attribute status record SHALL be included in the command, with the status field set to SUCCESS and the attribute identifier field omitted.

2.5.17 Write Attributes Structured Response Command

2.5.17.1 Write Attributes Structured Response Command Frame Format

The Write Attributes Response command frame SHALL be formatted as illustrated in Figure 2-34.

Figure 2-34. Write Attributes Structured Response Command Frame

Octets: Variable	**Variable**	**Variable**	**...**	**Variable**
ZCL header	Write attribute status record 1	Write attribute status record 2	...	Write attribute status record *n*

Each write attribute status record SHALL be formatted as illustrated in Figure 2-35.

Figure 2-35. Format of the Write Attribute Status Record Field

Octets: 1	**2**	**Variable**
Status	Attribute identifier	Selector

2.5.17.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Write Attributes defined for any cluster in the ZCL or 1 if this command is being used to write manufacturer specific attributes.

The command identifier field SHALL be set to indicate the Write Attributes Structured response command (see Table 2-3).

2.5.17.1.2 Status Field

The status field is 8 bits in length and specifies the status of the write operation attempted on this attribute, as detailed in Effect on Receipt2.5.16.3.

Note that write attribute status records are not included for successfully written attributes, to save bandwidth. In the case of successful writing of all attributes, only a single write attribute status record SHALL be included in the command, with the status field set to SUCCESS and the attribute identifier and selector fields omitted.

2.5.17.1.3 Attribute Identifier Field

The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute on which the write operation was attempted.

2.5.17.1.4 Selector Field

The selector field SHALL specify the element of the attribute on which the write operation that failed was attempted. See Figure 2-33 for the structure of this field.

From the structure shown in Figure 2-33, note that for all attribute data types other than array or structure this field consists of a single octet with value zero. For array or structure types, a single octet with value zero indicates that no information is available about which element of the attribute caused the failure.

2.5.17.2 When Generated

The Write Attributes Structured response command is generated in response to a Write Attributes Structured command.

2.5.17.3 Effect on Receipt

On receipt of this command, the device is notified of the results of its original Write Attributes Structured command.

2.5.18 Discover Commands Received Command

This command may be used to discover all commands processed (received) by this cluster, including optional or manufacturer-specific commands.

2.5.18.1 Discover Commands Received Command Frame Format

The discover server commands command frame SHALL be formatted as follows.

Figure 2-36. Format of the Discover Server Commands Command Frame

Octets:	Variable	1	1
Field:	ZCL header	Start command identifier	Maximum command identifiers

2.5.18.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer-specific sub-field SHALL be set to 0 to discover standard commands in a cluster or 1 to discover manufacturer-specific commands in either a standard or a manufacturer-specific cluster. A manufacturer ID in this field of 0xffff (wildcard) will discover any manufacture-specific commands. The direction bit SHALL be 0 (client to server) to discover commands that the server can process. The direction bit SHALL be 1 (server to client) to discover commands that the client can process.

The command identifier field SHALL be set to indicate the Discover Commands Received command.

2.5.18.1.2 Start Command Identifier Field

The start command identifier field is 8-bits in length and specifies the value of the identifier at which to begin the command discovery.

2.5.18.1.3 Maximum Command Identifiers Field

The maximum command identifiers field is 8-bits in length and specifies the maximum number of command identifiers that are to be returned in the resulting Discover Commands Received Response.

2.5.18.2 When Generated

The Discover Commands Received command is generated when a remote device wishes to discover the optional and mandatory commands the cluster to which this command is sent can process.

2.5.18.3 Effect on Receipt

On receipt of this command, the device SHALL construct an ordered list of command identifiers. This list SHALL start with the first command that has an identifier that is equal to or greater than the identifier specified in the start command identifier field. The number of command identifiers included in the list SHALL not exceed that specified in the maximum command identifiers field.

2.5.19 Discover Commands Received Response

The Discover Commands Received Response command is sent in response to a Discover Commands Received command, and is used to discover which commands a cluster can process.

2.5.19.1 Discover Commands Received Response Frame

The Discover Commands Received Response command frame SHALL be formatted as shown below:

Figure 2-37. Format of the Discover Commands Received Response Frame

Octets:	Variable	1	1	1	…	1
Field:	ZCL Header	Discovery complete	Command identifier 1	Command identifier 2	…	Command identifier n

2.5.19.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer-specific sub-field SHALL be set to the same value included in the original discover commands command, with the exception that if the manufacture ID is 0xffff (wildcard), then the response will contain the manufacture ID of the manufacturer-specific commands, or will not be present if the cluster supports no manufacturer-specific extensions, or the manufacturer wishes to hide the fact that it supports extensions. The command identifier field SHALL be set to indicate the Discover Commands Received Response command.

2.5.19.1.2 Discovery Complete Field

The discovery complete field is a boolean field. A value of 0 indicates that there are more commands to be discovered. A value of 1 indicates that there are no more commands to be discovered.

2.5.19.1.3 Command Identifier Field

The command identifier field SHALL contain the identifier of a discovered command. Commands SHALL be included in ascending order, starting with the lowest attribute identifier that is greater than or equal to the start attribute identifier field of the received discover server commands command.

2.5.19.2 When Generated

The Discover Commands Received Response is generated in response to a Discover Commands Received command.

2.5.19.3 Effect on Receipt

On receipt of this command, the device is notified of the results of its command discovery request. Following the receipt of this command, if the discovery complete field indicates that there are more commands to be discovered, the device may choose to send subsequent discover command request commands to obtain the rest of the command identifiers. In this case, the start command identifier specified in the next command discovery request command SHOULD be set equal to one plus the last command identifier received in the Discover Commands Received Response.

2.5.20 Discover Commands Generated Command

This command may be used to discover all commands which may be generated (sent) by the cluster, including optional or manufacturer-specific commands.

2.5.20.1 Discover Commands Generated Command Frame Format

Except for the command ID in the ZCL header, the Discover Commands Generated command frame SHALL be formatted as described in sub-clause 2.5.18 and its subsections.

2.5.20.2 When Generated

The Discover Commands Generated command is generated when a remote device wishes to discover the commands that a cluster may generate on the device to which this command is directed.

2.5.20.3 Effect on Receipt

On receipt of this command, the device SHALL construct an ordered list of command identifiers. This list SHALL start with the first command that has an identifier that is equal to or greater than the identifier specified in the start command identifier field. The number of command identifiers included in the list SHALL not exceed that specified in the maximum command identifiers field.

2.5.21 Discover Commands Generated Response

The Discover Commands Generated Response command is sent in response to a Discover Commands Generated command, and is used to discover which commands a cluster supports.

2.5.21.1 Discover Commands Generated Response Frame

Except for the command ID in the ZCL header, the Discover Commands Generated Response command frame SHALL be formatted as described in sub-clause 2.5.18 and its subsections.

2.5.21.2 When Generated

The Discover Commands Generated Response is generated in response to a Discover Commands Generated command.

2.5.21.3 Effect on Receipt

On receipt of this command, the device is notified of the results of its Discover Commands Generated command.

Following the receipt of this command, if the discovery complete field indicates that there are more commands to be discovered, the device may choose to send subsequent Discover Commands Generated commands to obtain the rest of the command identifiers. In this case, the start command identifier specified in the next Discover Commands Generated command SHOULD be set equal to one plus the last command identifier received in the Discover Commands Generated Response.

2.5.22 Discover Attributes Extended Command

This command is similar to the discover attributes command, but also includes a field to indicate whether the attribute is readable, writeable or reportable.

2.5.22.1 Discover Attributes Extended Command Frame Format

The Discover Attributes Extended command frame SHALL be formatted as illustrated as follows.

Figure 2-38. Format of the Discover Attributes Extended Command Frame

Octets:	Variable	2	1
Field:	ZCL Header	Start attribute identifier	Maximum attribute identifiers

2.5.22.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer-specific sub-field SHALL be set to 0 to discover standard attributes in a cluster or 1 to discover manufacturer-specific attributes in either a standard or a manufacturer-specific cluster. A manufacturer ID in this field of 0xffff (wildcard) will discover any manufacture-specific attributes. The direction bit SHALL be 0 (client to server) to discover attributes that the server hosts. The direction bit SHALL be 1 (server to client) to discover attributes that the client may host.

The command identifier field SHALL be set to indicate the Discover Attributes Extended command.

2.5.22.1.2 Start Attribute Identifier Field

The start attribute identifier field is 16-bits in length and specifies the value of the identifier at which to begin the attribute discovery.

2.5.22.1.3 Maximum Attribute Identifiers Field

The maximum attribute identifiers field is 8 bits in length and specifies the maximum number of attribute identifiers that are to be returned in the resulting Discover Attributes Extended Response command.

2.5.22.2 When Generated

The Discover Attributes Extended command is generated when a remote device wishes to discover the identifiers and types of the attributes on a device which are supported within the cluster to which this command is directed, including whether the attribute is readable, writeable or reportable.

2.5.22.3 Effect on Receipt

On receipt of this command, the device SHALL construct an ordered list of attribute information records, each containing a discovered attribute identifier and its data type, in ascending order of attribute identifiers. This list SHALL start with the first attribute that has an identifier that is equal to or greater than the identifier specified in the start attribute identifier field. The number of attribute identifiers included in the list SHALL not exceed that specified in the maximum attribute identifiers field.

2.5.23 Discover Attributes Extended Response Command

This command is sent in response to a Discover Attributes Extended command, and is used to determine if attributes are readable, writable or reportable.

2.5.23.1 Discover Attributes Extended Response Command Frame Format

The Discover Attributes Extended Response command frame SHALL be formatted as illustrated as follows.

Figure 2-39. Format of the Discover Attributes Extended Response Command Frame

Octets:	Variable	1	4	4	…	4
Field:	ZCL header	Discovery complete	Extended attribute information 1	Extended attribute information 2	…	Extended attribute information n

Each extended attribute information field SHALL be formatted as follows.

Figure 2-40. Format of the Extended Attribute Information Fields

Octets:	2	1	1
Field:	Attribute identifier	Attribute data type	Attribute access control

2.5.23.1.1 ZCL Header Fields

The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global command (0b00). The manufacturer-specific sub-field SHALL be set to the same value included in the original Discover Attributes Extended command, with the exception that if the manufacture ID is 0xffff (wildcard), then the response will contain the manufacture ID of the manufacturer-specific attributes, or will not be present if the cluster supports no manufacturer-specific extensions.

The command identifier field SHALL be set to indicate the Discover Attributes Extended Response command.

2.5.23.1.2 Discovery Complete Field

The discovery complete field is a boolean field. A value of 0 indicates that there are more attributes to be discovered. A value of 1 indicates that there are no more attributes to be discovered.

2.5.23.1.3 Attribute Identifier Field

The attribute identifier field SHALL contain the identifier of a discovered attribute. Attributes SHALL be included in ascending order, starting with the lowest attribute identifier that is greater than or equal to the start attribute identifier field of the received discover attributes command.

2.5.23.1.4 Attribute Data Type Field

The attribute data type field SHALL contain the data type of the attribute.

2.5.23.1.5 Attribute Access Control Field

The attribute access control field SHALL indicate whether the attribute is readable, writable, and/or reportable. This is an 8-bit bitmask field as shown below: The bits are in little endian order (bit 0 is listed first).

Figure 2-41. Format of the Attribute Access Control Field

Bits:	**1**	1	1
Field:	**Readable**	Writeable	Reportable

2.5.23.2 When Generated

The Discover Attributes Extended Response command is generated in response to a Discover Attributes Extended command.

2.5.23.3 Effect on Receipt

On receipt of this command, the device is notified of the results of its Discover Attributes Extended command.

Following the receipt of this command, if the discovery complete field indicates that there are more attributes to be discovered, the device may choose to send subsequent Discover Attributes Extended commands to obtain the rest of the attribute identifiers and access control. In this case, the start attribute identifier specified in the next Discover Attributes Extended command SHOULD be set equal to one plus the last attribute identifier received in the Discover Attributes Extended Response command.

2.6 Addressing, Types and Enumerations

2.6.1 Addressing

The architecture uses a number of concepts to address applications, clusters, device descriptions, attributes and commands, each with their own constraints. This sub-clause details these constraints.

2.6.1.1 Profile Identifier

A profile identifier is 16 bits in length and specifies the application profile being used. A profile identifier SHALL be set to one of the non-reserved values listed in Table 2-5. Please see [Z5], Application Architecture for more details.

Table 2-5. Valid Profile Identifier Values

Profile Identifier	Description
0x0000 – 0x7fff	Standard application profile [Z7]
0xc000 – 0xffff	Manufacturer Specific application profile
all other values	Reserved

2.6.1.2 Device Identifier

A device identifier is 16 bits in length and specifies a specific device within a standard. A device identifier SHALL be set to one of the non-reserved values listed in Table 2-6. Please see [Z5], Application Architecture for more details.

Table 2-6. Valid Device Identifier Values

Device Identifier	Description
0x0000 – 0xbfff	Standard device description.
all other values	Reserved

2.6.1.3 Cluster Identifier

A cluster identifier is 16 bits in length and identifies an instance of an implemented cluster specification (see 2.2.1.1). It SHALL be set to one of the non-reserved values listed in Table 2-7. Please see [Z5], Application Architecture for more details.

Table 2-7. Valid Cluster Identifier Values

Cluster Identifier	Description
0x0000 – 0x7fff	Standard cluster
0xfc00 – 0xffff	Manufacturer specific cluster
all other values	Reserved

2.6.1.4 Attribute Identifier

An attribute identifier is 16 bits in length and specifies a single attribute within a cluster. An attribute identifier, defined within the ZCL, SHALL be set to one of the non-reserved values listed in Table 2-8. Undefined cluster attributes are reserved for future cluster attributes. Global attributes are associated with all clusters (see 2.3.4.3).

Table 2-8. Valid ZCL Defined Attribute Identifier Values

Attribute Identifier	Description
0x0000 – 0x4fff	Standard attribute
0xf000 – 0xfffe	Global Attributes
all other values	Reserved

Manufacturer specific attributes within a standard cluster can be defined over the full 16-bit range. These may be manipulated using the global commands listed in Table 2-3, with the frame control field set to indicate a manufacturer specific command (see 2.4). (Note that, alternatively, the manufacturer may define his own cluster specific commands (see 2.4), re-using these command IDs if desired).

2.6.1.5 Command Identifier

A command identifier is 8 bits in length and specifies a global command within the ZCL or a cluster specific command. A command identifier SHALL be set to one of the non-reserved values listed in Table 2-9. Manufacturer specific commands within a standard cluster can be defined over the full 8-bit range but each SHALL use the appropriate manufacturer code.

Table 2-9. Valid ZCL-Defined Command Identifier Values

Command Identifier	Description
0x00 – 0x7f	Standard command or Manufacture Specific command, depending on the Frame Control field in the ZCL Header
all other values	Reserved

2.6.2 Data Types

Each attribute and command field in a cluster specification SHALL have a well-defined data type. Each attribute in a cluster specification SHALL map to a single data type identifier (data type ID), which describes the length and general properties of the data type.

New data type identifiers SHALL NOT be added to this table. It is encouraged that commonly used cluster attribute and command field data types are added to this list and mapped to appropriate data type identifiers with a unique name. Such common data types can then be reused instead of redefined in each specification. For example: a percentage data type representing 0-100% with a unit size of .5 percent, would be mapped to data type identifier 0x20 (also unsigned 8-bit integer), and perhaps named ‘Percent 8-bit .5-unit’ (short named ‘percent8.5’).

The table also indicates for each data type whether it defines an analog or discrete value. Values of analog types may be added to or subtracted from other values of the same type, and are typically used to measure the value of properties in the real world that vary continuously over a range. Values of discrete data types only have meaning as individual values, and may not be added or subtracted.

Cluster specifications SHALL use the unique data type short name to reduce the text size of the specification.

The Invalid Value is only specified when an invalid data value is required for an attribute. The Invalid Value for an analog data type MAY be within the valid data range as defined in an attribute specification.[6]

Table 2-10. Data Types

Class	Data Type	Short	ID	Length (Octets)	Invalid Value
Null	No data	nodata	0x00	0	-
General data Discrete	8-bit data	data8	0x08	1	-
	16-bit data	data16	0x09	2	-
	24-bit data	data24	0x0a	3	-
	32-bit data	data32	0x0b	4	-
	40-bit data	data40	0x0c	5	-
	48-bit data	data48	0x0d	6	-
	56-bit data	data56	0x0e	7	-
	64-bit data	data64	0x0f	8	-
Logical Discrete	Boolean	bool	0x10	1	0xff
Bitmap Discrete	8-bit bitmap	map8	0x18	1	-
	16-bit bitmap	map16	0x19	2	-
	24-bit bitmap	map24	0x1a	3	-
	32-bit bitmap	map32	0x1b	4	-

[6] CCB 2213 explain about invalid values

Class	Data Type	Short	ID	Length (Octets)	Invalid Value
	40-bit bitmap	map40	0x1c	5	-
	48-bit bitmap	map48	0x1d	6	-
	56-bit bitmap	map56	0x1e	7	-
	64-bit bitmap	map64	0x1f	8	-
Unsigned integer Analog	Unsigned 8-bit integer	uint8	0x20	1	0xff
	Unsigned 16-bit integer	uint16	0x21	2	0xffff
	Unsigned 24-bit integer	uint24	0x22	3	0xffffff
	Unsigned 32-bit integer	uint32	0x23	4	0xffffffff
	Unsigned 40-bit integer	uint40	0x24	5	0xffffffffff
	Unsigned 48-bit integer	uint48	0x25	6	0xffffffffffff
	Unsigned 56-bit integer	uint56	0x26	7	0xffffffffffffff
	Unsigned 64-bit integer	uint64	0x27	8	0xffffffffffffffff
Signed integer Analog	Signed 8-bit integer	int8	0x28	1	0x80
	Signed 16-bit integer	int16	0x29	2	0x8000
	Signed 24-bit integer	int24	0x2a	3	0x800000
	Signed 32-bit integer	int32	0x2b	4	0x80000000
	Signed 40-bit integer	int40	0x2c	5	0x8000000000
	Signed 48-bit integer	int48	0x2d	6	0x800000000000
	Signed 56-bit integer	int56	0x2e	7	0x80000000000000
	Signed 64-bit integer	int64	0x2f	8	0x8000000000000000
Enumeration Discrete	8-bit enumeration	enum8	0x30	1	0xff
	16-bit enumeration	enum16	0x31	2	0xffff
Floating point	Semi-precision	semi	0x38	2	Not a Number

Class	Data Type	Short	ID	Length (Octets)	Invalid Value
Analog	Single precision	single	0x39	4	Not a Number
	Double precision	double	0x3a	8	Not a Number
String Discrete	Octet string	octstr	0x41	Defined in first octet	0xff in first octet
	Character string	string	0x42	Defined in first octet	0xff in first octet
	Long octet string	octstr16	0x43	Defined in first two octets	0xffff in first two octets
	Long character string	string16	0x44	Defined in first two octets	0xffff in first two octets
	Fixed ASCII	ASCII	-	defined in specification	-
Ordered sequence Discrete	Array	array	0x48	2 + sum of lengths of contents	0xffff in first 2 octets
	Structure	struct	0x4c	2 + sum of lengths of contents	0xffff in first 2 octets
Collection Discrete	Set	set	0x50	Sum of lengths of contents	Number of elements returned as 0xffff
	Bag	bag	0x51	Sum of lengths of contents	Number of elements returned as 0xffff
Time Analog	Time of day	ToD	0xe0	4	0xffffffff
	Date	date	0xe1	4	0xffffffff
	UTCTime	UTC	0xe2	4	0xffffffff
Identifier Discrete	Cluster ID	clusterId	0xe8	2	0xffff
	Attribute ID	attribId	0xe9	2	0xffff
	BACnet OID	bacOID	0xea	4	0xffffffff
Miscellaneous Discrete	IEEE address	EUI64	0xf0	8	0xffffffffffffffff
	128-bit security key	key128	0xf1	16	-

Class	Data Type	Short	ID	Length (Octets)	Invalid Value
	Opaque	opaque	-	fixed or defined separately	
Unknown	Unknown	unk	0xff	0	-

2.6.2.1 No Data Type

The no data type is a special type to represent an attribute with no associated data.

2.6.2.2 General Data (8, 16, 24, 32, 40, 48, 56 and 64-bit)

This type has no rules about its use, and may be used when a data element is needed but its use does not conform to any of the standard types.

2.6.2.3 Boolean

The Boolean type represents a logical value, either FALSE (0x00) or TRUE (0x01). The value 0xff represents an invalid value of this type. All other values of this type are forbidden.

2.6.2.4 Bitmap (8, 16, 24, 32, 40, 48, 56 and 64-bit)

The Bitmap type holds 8, 16, 24, 32, 40, 48, 56 or 64 logical values, one per bit, depending on its length. There is no value that represents an invalid value of this type.

2.6.2.5 Unsigned Integer (8, 16, 24, 32, 40, 48, 56 and 64-bit)

This type represents an unsigned integer with a decimal range of 0 to 2^8-1, 0 to 2^{16}-1, 0 to 2^{24}-1, 0 to 2^{32}-1, 0 to 2^{40}-1, 0 to 2^{48}-1, 0 to 2^{56}-1, or 0 to 2^{64}-1, depending on its length. The values that represent an invalid value of this type are 0xff, 0xffff, 0xffffff, 0xffffffff, 0xffffffffff, 0xffffffffffff, 0xffffffffffffff and 0xffffffffffffffff respectively.

2.6.2.6 Signed Integer (8, 16, 24, 32, 40, 48, 56 and 64-bit)

This type represents a signed integer with a decimal range of -(2^7-1) to 2^7-1, -(2^{15}-1) to 2^{15}-1, -(2^{23}-1) to 2^{23}-1, -(2^{31}-1) to 2^{31}-1, -(2^{39}-1) to 2^{39}-1, -(2^{47}-1) to 2^{47}-1, -(2^{55}-1) to 2^{55}-1, or -(2^{63}-1) to 2^{63}-1, depending on its length. The values that represent an invalid value of this type are 0x80, 0x8000, 0x800000, 0x80000000, 0x8000000000, 0x800000000000, 0x80000000000000 and 0x8000000000000000 respectively.

2.6.2.7 Enumeration (8-bit, 16-bit)

The Enumeration type represents an index into a lookup table to determine the final value. The values 0xff and 0xffff represent invalid values of the 8-bit and 16-bit types respectively.

2.6.2.8 Semi-precision

The semi-precision number format is based on the IEEE 754 standard for binary floating-point arithmetic [E2]. This number format SHOULD be used very sparingly, when necessary, keeping in mind the code and processing required supporting it.

The value is calculated as:

$$\text{Value} = -1^{\text{Sign}} * (\text{Hidden} + \text{Mantissa}/1024) * 2^{(\text{Exponent}-15)}$$

Figure 2-42. Format of the Semi-precision Number

	Sign	Exponent					Hidden		Mantissa									
	S	E_4	E_3	E_2	E_1	E_0	H	.	M_9	M_8	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0
bit	15					10			9									0

Note: The transmission order for the format in Figure 2-42 is bit 0 first.

For normalized numbers ($>2^{-14}$), the hidden bit = 1 and the resolution is constant at 11 bits (1 in 2048).

For un-normalized numbers, the hidden bit = 0. Note that this does not maintain 11-bit resolution and that the resolution becomes coarser as the number gets smaller.

The hidden bit is not sent over the link. It SHALL have the value '1' (i.e., normalized) in order to be classified as a semi-precision number.

The sign bit is set to 0 for positive values, 1 for negative.

The exponent is 5 bits. The actual exponent of 2 is calculated as (exponent – 15).

Certain values are reserved for specific purposes:

- **Not a Number**: this is used for undefined values (e.g., at switch-on and before initialization) and is indicated by an exponent of 31 with a non-zero mantissa.
- **Infinity:** this is indicated by an exponent of 31 and a zero mantissa. The sign bit indicates whether this represents + infinity or – infinity, the figure of 0x7c00 representing $+\infty$ and 0xfc00 representing $-\infty$.
- **Zero:** this is indicated by both a zero exponent and zero mantissa. The sign bit indicates whether this is + or – zero, the value 0x0000 representing +zero and 0x8000 representing –zero.
- **Un-normalized numbers:** numbers $< 2^{-14}$ are indicated by a value of 0 for the exponent. The hidden bit is set to zero.

The maximum value represented by the mantissa is 0x3ff / 1024. The largest number that can be represented is therefore:

$$-1^{\text{Sign}} * (1 + 1023/1024) * 2^{(30-15)} = \pm 1.9990234 * 32768 = \pm 65504$$

Certain applications may choose to scale this value to allow representation of larger values (with a correspondingly coarser resolution). For details, see the relevant device descriptions.

For example, a value of +2 is represented by $+2^{(16-15)} * 1.0$ = 0x4000, while a value of –2 is represented by 0xc000.

Similarly, a value of +0.625 is represented by $+2^{(17-15)} * 1.625$ = 0x4680, while –0.625 is represented by 0xc680.

2.6.2.9 Single Precision

The format of the single precision data type is based on the IEEE 754 standard for binary floating-point arithmetic [E2]. This number format SHOULD be used very sparingly, when necessary, keeping in mind the code and processing required supporting it.

The format and interpretation of values of this data type follow the same rules as given for the semi-precision data type, but with longer sub-fields, as follows.

Length of mantissa = 23 bits, length of exponent = 8 bits

For further details, see [E2].

2.6.2.10 Double Precision

The format of the double precision data type is based on the IEEE 754 standard for binary floating-point arithmetic [E2]. This number format SHOULD be used very sparingly, when necessary, keeping in mind the code and processing required supporting it.

The format and interpretation of values of this data type follow the same rules as given for the semi-precision data type, but with longer sub-fields, as follows.

Length of mantissa = 52 bits, length of exponent = 11 bits

For further details, see [E2].

2.6.2.11 Octet String

The octet string data type contains data in an application-defined format, not defined in this specification. The octet string data type is formatted as illustrated in Figure 2-43.

Figure 2-43. Format of the Octet String Type

Octets: 1	Variable
Octet count	Octet data

The octet count sub-field is one octet in length and specifies the number of octets contained in the octet data sub-field.

Setting this sub-field to 0x00 represents an octet string with no octet data (an “empty string”). Setting this sub-field to 0xff represents an invalid octet string value. In both cases the octet data sub-field has zero length.

The octet data sub-field is *n* octets in length, where *n* is the value of the octet count sub-field. This sub-field contains the application-defined data.

2.6.2.12 Character String

The character string data type contains data octets encoding characters according to the language and character set field of the complex descriptor (see [Z1]). If not specified by the complex descriptor, the default character encoding SHALL be UTF-8. The character string data type SHALL be formatted as illustrated in Figure 2-44.

Figure 2-44. Format of the Character String Type

Octets: 1	Variable
Character data length	Character data

The character data length sub-field is one octet in length and specifies the length of the character data sub-field. (**Note:** for the ISO 646 ASCII character set, this is the same as the number of characters in the string. For other codings, this may not be the case.)

Setting this sub-field to 0x00 represents a character string with no character data (an “empty string”). Setting this sub-field to 0xff represents an invalid character string value. In both cases the character data sub-field has zero length.

The character data sub-field contains the encoded characters that comprise the desired character string. Its length is the sum of the lengths of the characters as specified by the language and character set fields of the complex descriptor.

A character string with no contents, i.e., with the character count sub-field equal to 0x00 and a zero length character data sub-field, SHALL be referred to as an 'empty string'.

2.6.2.13 Long Octet String

The long octet string data type contains data in an application-defined format, not defined in this specification. The long octet string data type is formatted as illustrated in Figure 2-45.

Figure 2-45. Format of the Long Octet String Type

Octets: 2	Variable
Octet count	Octet data

The octet count sub-field is two octets in length and specifies the number of octets contained in the octet data sub-field. It has the same format as a 16-bit unsigned integer (see 2.6.2.5).

Setting this sub-field to 0x0000 represents a long octet string with no octet data (an “empty string”). Setting this sub-field to 0xffff represents an invalid long octet string value. In both cases the octet data sub-field has zero length.

The octet data sub-field is *n* octets in length, where *n* is the value of the octet count sub-field. This sub-field contains the application-defined data.

2.6.2.14 Long Character String

The long character string data type contains data octets encoding characters according to the language and character set field of the complex descriptor (see [Z1]). If not specified by the complex descriptor, the default character encoding SHALL be UTF-8. The long character string data type is formatted as illustrated in Figure 2-46.

Figure 2-46. Format of the Long Character String Type

Octets: 2	Variable
Character count	Character data

The character count sub-field is two octets in length and specifies the length of the character data sub-field. (**Note:** for the ISO 646 ASCII character set, this is the same as the number of characters in the string. For other codings, this may not be the case.) It has the same format as a 16-bit unsigned integer (see 2.6.2.5).

Setting this sub-field to 0x0000 represents a long character string with no character data (an “empty string”). Setting this sub-field to 0xffff represents an invalid long character string value. In both cases the character data sub-field has zero length.

The character data sub-field contains the encoded characters that comprise the desired character string. Its length is the sum of the lengths of the characters as specified by the language and character set fields of the complex descriptor.

A character string with no contents, i.e., with the character count sub-field equal to 0x0000 and a zero length character data sub-field, SHALL be referred to as an 'empty string'.

2.6.2.15 Fixed ASCII

This data type is defined for legacy reasons, so that there is a data type to represent an ASCII display string that has a fixed length as defined in the specification. The NUL ASCII character 0x00 shall terminate the string, unless the string takes up the entire fixed length.

This data type SHALL NOT be used as a data type for an attribute, because it does not have an associated length, nor Data Type Id. It is not recommended to use this data type, when a more well-defined data type exists.

2.6.2.16 Array

An array is an ordered sequence of zero or more elements, all of the same data type. This data type may be any ZCL defined data type, including array, structure, bag or set. The total nesting depth is limited to 15, and may be further limited by an application.

Individual elements may be accessed by an index of type 16-bit unsigned integer. Elements are numbered from 1 upwards. The zeroth element is readable, always has type uint16, and holds the number of elements contained in the array, which may be zero. If the zeroth element contains 0xffff, the array is considered invalid / undefined.

The zeroth element may also, as an implementation option, be writeable, to change the size of the array (see 2.5.16.1 for details).

Arrays are 'packed', i.e., there is no concept of a 'null' element. However, if an element has a simple (unstructured) type, and that type has an 'invalid number' value defined, then that value indicates that the element is invalid / undefined.

2.6.2.17 Structure

A structure is an ordered sequence of elements, which may be of different data types. Each data type may be any ZCL defined data type, including array, structure, bag or set. The total nesting depth is limited to 15, and may be further limited by an application.

Individual elements may be accessed by an index of type 16-bit unsigned integer. Elements are numbered from 1 upwards. The zeroth element is readable, always has type 16-bit unsigned integer, and holds the number of elements contained in the structure, which may be zero. If the zeroth element contains 0xffff, the array is considered invalid / undefined. The zeroth element may not be written to.

Structures are 'packed', i.e., there is no concept of a 'null' element. However, if an element has a simple (unstructured) type, and that type has an 'invalid number' value defined, that value indicates that the element is undefined.

2.6.2.18 Set

A set is a collection of elements with no associated order. Each element has the same data type, which may be any ZCL defined data type, including array, structure, bag or set. The nesting depth is limited to 15, and may be further limited by an application.

Elements of a set are not individually addressable, so may not be individually read or modified. Sets may only be read in their entirety. Individual elements may be added to a set or removed from a set; removal is done by value.

The maximum number of elements in a set is 0xfffe. If the number of elements is returned by a read command as 0xffff, this indicates that it is invalid / undefined.

No two elements of a set may have the same value.

2.6.2.19 Bag

A bag behaves the same as a set, except that the restriction that no two elements may have the same value is removed.

2.6.2.20 Time of Day

The Time of Day data type SHALL be formatted as illustrated in Figure 2-47.

Figure 2-47. Format of the Time of Day Type

Octets: 1	1	1	1
Hours	Minutes	Seconds	Hundredths

The hours subfield represents hours according to a 24-hour clock. The range is from 0 to 23.

The minutes subfield represents minutes of the current hour. The range is from 0 to 59.

The seconds subfield represents seconds of the current minute. The range is from 0 to 59.

The hundredths subfield represents 100ths of the current second. The range is from 0 to 99.

A value of 0xff in any subfield indicates an unused subfield. If all subfields have the value 0xff, this indicates an invalid or 'don't care' value of the data type.

2.6.2.21 Date

The Date data type SHALL be formatted as illustrated in Figure 2-48.

Figure 2-48. Format of the Date Type

Octets: 1	1	1	1
Year - 1900	Month	Day of month	Day of week

The year - 1900 subfield has a range of 0 to 255, representing years from 1900 to 2155.

The month subfield has a range of 1 to 12, representing January to December.

The day of month subfield has a range of 1 to 31. Note that values in the range 29 to 31 may be invalid, depending on the month and year.

The day of week subfield has a range of 1 to 7, representing Monday to Sunday.

A value of 0xff in any subfield indicates an unused subfield. If all subfields have the value 0xff, this indicates an invalid or 'don't care' value of the data type.

2.6.2.22 UTCTime

UTCTime is an unsigned 32-bit value representing the number of seconds since 0 hours, 0 minutes, 0 seconds, on the 1st of January, 2000 UTC (Universal Coordinated Time). The value that represents an invalid value of this type is 0xffffffff.

Note that UTCTime does not hold a standard textual representation of Universal Coordinated Time (UTC). However, UTC (to a precision of one second) may be derived from it.

2.6.2.23 Cluster ID

This type represents a cluster identifier as defined in 2.6.1.3.

2.6.2.24 Attribute ID

This type represents an attribute identifier as defined in 2.6.1.4.

2.6.2.25 BACnet OID (Object Identifier)

The BACnet OID data type is included to allow interworking with BACnet (see [A1]). The format is described in the referenced standard.

2.6.2.26 IEEE Address

The IEEE Address data type is a 64-bit IEEE address that is unique to every node. A value of 0xffffffffffffffff indicates that the address is unknown.

2.6.2.27 128-bit Security Key

The 128-bit Security Key data type may take any 128-bit value.

2.6.2.28 Opaque

Fixed block or series of octets where the length is determined separately. The length SHALL be fixed in the specification or determined from information from another part of the protocol. The format of the data MAY also be unknown. It is not recommended to use this data type, when a more well-defined data type exists.

This data type SHALL NOT be used as a cluster attribute, or have a Data Type Id.

2.6.2.29 Unknown

This data type SHALL NOT be used for a cluster attribute or frame data field. This is not an actual data type. It is listed here for completeness and to reserve the data type identifier for use where one is required to designate that a data type is unknown. It SHALL never be used to identify actual data as unknown. If the structure, format, or length of data is unknown, or an existing data type cannot be used, then the Opaque data type SHALL be used.

2.6.3 Status Enumerations

Where a ZCL command contains a status field, the actual value of the enumerated status values is listed in Table 2-11.

Table 2-11. Enumerated Status Values Used in the ZCL

Enumerated Status	Value	Description
SUCCESS	0x00	Operation was successful.
FAILURE	0x01	Operation was not successful.
NOT_AUTHORIZED	0x7e	The sender of the command does not have authorization to carry out this command.
reserved[7]	0x7f	
MALFORMED_COMMAND	0x80	The command appears to contain the wrong fields, as detected either by the presence of one or more invalid field entries or by there being missing fields. Command not carried out. Implementer has discretion as to whether to return this error or INVALID_FIELD.

[7] CCB 2318 a reserved field may be used in the future and then be non-zero

Enumerated Status	Value	Description
UNSUP_CLUSTER_COMMAND	0x81	The specified cluster command is not supported on the device. Command not carried out.
UNSUP_GENERAL_COMMAND	0x82	The specified general ZCL command is not supported on the device.
UNSUP_MANUF_CLUSTER_COMMAND	0x83	A manufacturer specific unicast, cluster specific command was received with an unknown manufacturer code, or the manufacturer code was recognized but the command is not supported.
UNSUP_MANUF_GENERAL_COMMAND	0x84	A manufacturer specific unicast, ZCL specific command was received with an unknown manufacturer code, or the manufacturer code was recognized but the command is not supported.
INVALID_FIELD	0x85	At least one field of the command contains an incorrect value, according to the specification the device is implemented to.
UNSUPPORTED_ATTRIBUTE	0x86	The specified attribute does not exist on the device.
INVALID_VALUE	0x87	Out of range error, or set to a reserved value. Attribute keeps its old value. Note that an attribute value may be out of range if an attribute is related to another, e.g., with minimum and maximum attributes. See the individual attribute descriptions for specific details.
READ_ONLY	0x88	Attempt to write a read-only attribute.
INSUFFICIENT_SPACE	0x89	An operation failed due to an insufficient amount of free space available.
DUPLICATE_EXISTS	0x8a	An attempt to create an entry in a table failed due to a duplicate entry already being present in the table.
NOT_FOUND	0x8b	The requested information (e.g., table entry) could not be found.
UNREPORTABLE_ATTRIBUTE	0x8c	Periodic reports cannot be issued for this attribute.
INVALID_DATA_TYPE	0x8d	The data type given for an attribute is incorrect. Command not carried out.
INVALID_SELECTOR	0x8e	The selector for an attribute is incorrect.
WRITE_ONLY	0x8f	A request has been made to read an attribute that the requestor is not authorized to read. No action taken.
INCONSISTENT_STARTUP_STATE	0x90	Setting the requested values would put the device in an inconsistent state on startup. No action taken.
DEFINED_OUT_OF_BAND	0x91	An attempt has been made to write an attribute that is present but is defined using an out-of-band method and not over the air.

Enumerated Status	Value	Description
INCONSISTENT	0x92	The supplied values (e.g., contents of table cells) are inconsistent.
ACTION_DENIED	0x93	The credentials presented by the device sending the command are not sufficient to perform this action.
TIMEOUT	0x94	The exchange was aborted due to excessive response time.
ABORT	0x95	Failed case when a client or a server decides to abort the upgrade process.
INVALID_IMAGE	0x96	Invalid OTA upgrade image (ex. failed signature validation or signer information check or CRC check).
WAIT_FOR_DATA	0x97	Server does not have data block available yet.
NO_IMAGE_AVAILABLE	0x98	No OTA upgrade image available for the client.
REQUIRE_MORE_IMAGE	0x99	The client still requires more OTA upgrade image files to successfully upgrade.
NOTIFICATION_PENDING	0x9a	The command has been received and is being processed.
HARDWARE_FAILURE	0xc0	An operation was unsuccessful due to a hardware failure.
SOFTWARE_FAILURE	0xc1	An operation was unsuccessful due to a software failure.
CALIBRATION_ERROR	0xc2	An error occurred during calibration.
UNSUPPORTED_CLUSTER	0xc3	The cluster is not supported
LIMIT_REACHED	0xc4	Limit of attribute range reached. Value is trimmed to closest limit (maximum or minimum).

CHAPTER 3 GENERAL

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

3.1 General Description

3.1.1 Introduction

The clusters specified in this document are generic measurement and sensing interfaces that are sufficiently general to be of use across a wide range of application domains.

3.1.2 Cluster List

This section lists the clusters specified in this document, and gives examples of typical usage.

Table 3-1. Device Configuration and Installation Clusters

ID	Cluster Name	Description
0x0000	Basic	Attributes for determining basic information about a device, setting user device information such as description of location, and enabling a device.
0x0001	Power Configuration	Attributes for determining more detailed information about a device's power source(s), and for configuring under/over voltage alarms.
0x0002	Device Temperature Configuration	Attributes for determining information about a device's internal temperature, and for configuring under/over temperature alarms.
0x0003	Identify	Attributes and commands for putting a device into Identification mode (e.g., flashing a light)

Figure 3-1. Typical Usage of Device Configuration and Installation Clusters

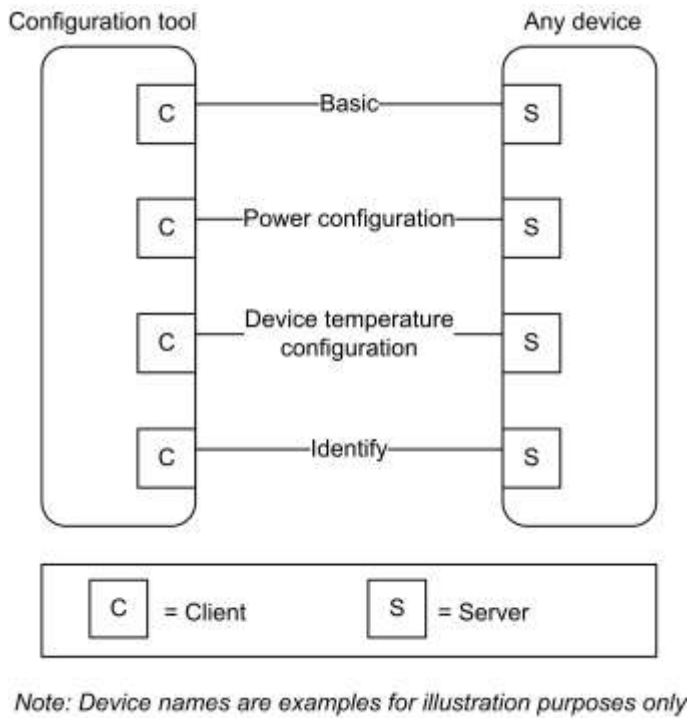

Configuration tool
Any device
C
Basic
S
C
Power configuration
S
C
Device temperature configuration
S
C
Identify
S
C = Client
S = Server
Note: Device names are examples for illustration purposes only

Table 3-2. Groups and Scenes Clusters

ID	Name	Description
0x0004	Groups	Attributes and commands for allocating a device to one or more of a number of groups of devices, where each group is addressable by a group address.
0x0005	Scenes	Attributes and commands for setting up and recalling a number of scenes for a device. Each scene corresponds to a set of stored values of specified device attributes.

Table 3-3. On/Off and Level Control Clusters

ID	Name	Description
0x0006	On/Off	Attributes and commands for switching devices between ‘On’ and ‘Off’ states.
0x0007	On/Off Switch Configuration	Attributes and commands for configuring on/off switching devices
0x0008	Level Control	Attributes and commands for controlling a characteristic of devices that can be set to a level between fully ‘On’ and fully ‘Off’.

Figure 3-2. Typical Usage of On/Off and Level Control Clusters

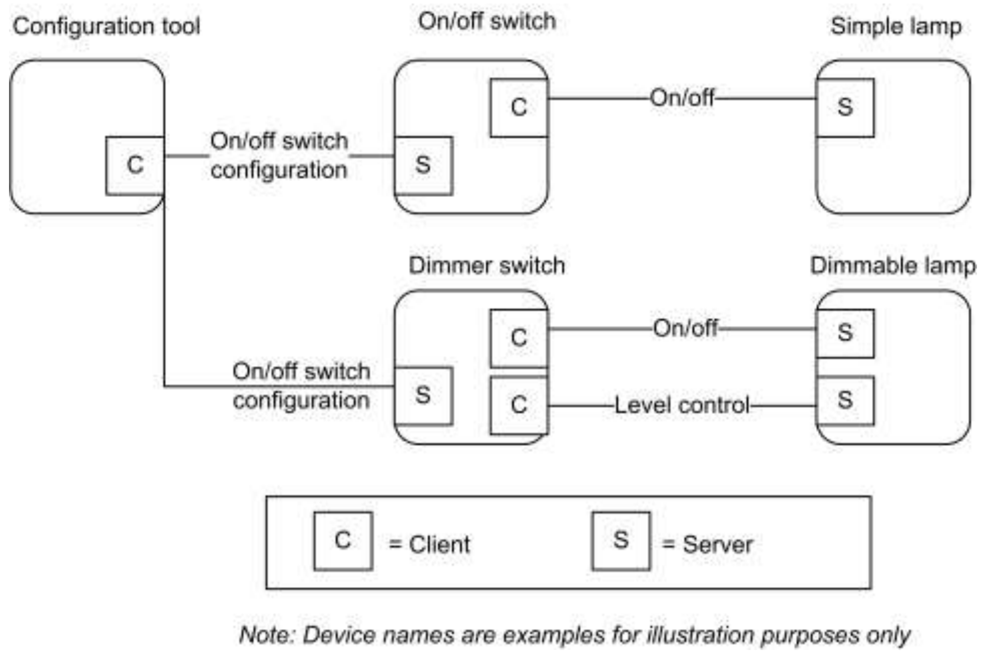
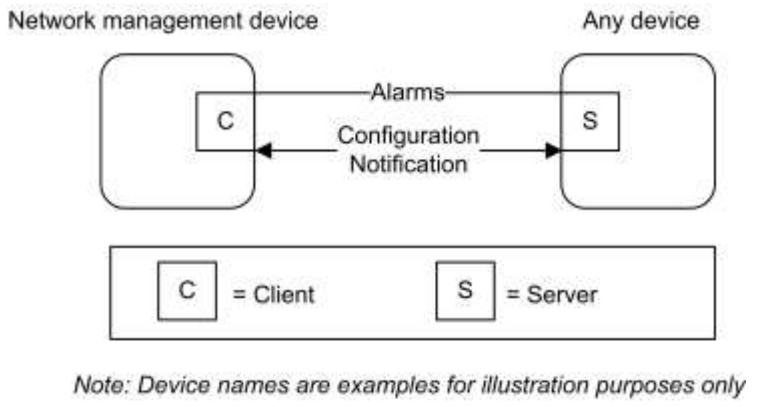


Table 3-4. Alarms Cluster

ID	Name	Description
0x0009	Alarms	Attributes and commands for sending alarm notifications and configuring alarm functionality.

Figure 3-3. Typical Usage of the Alarms Cluster

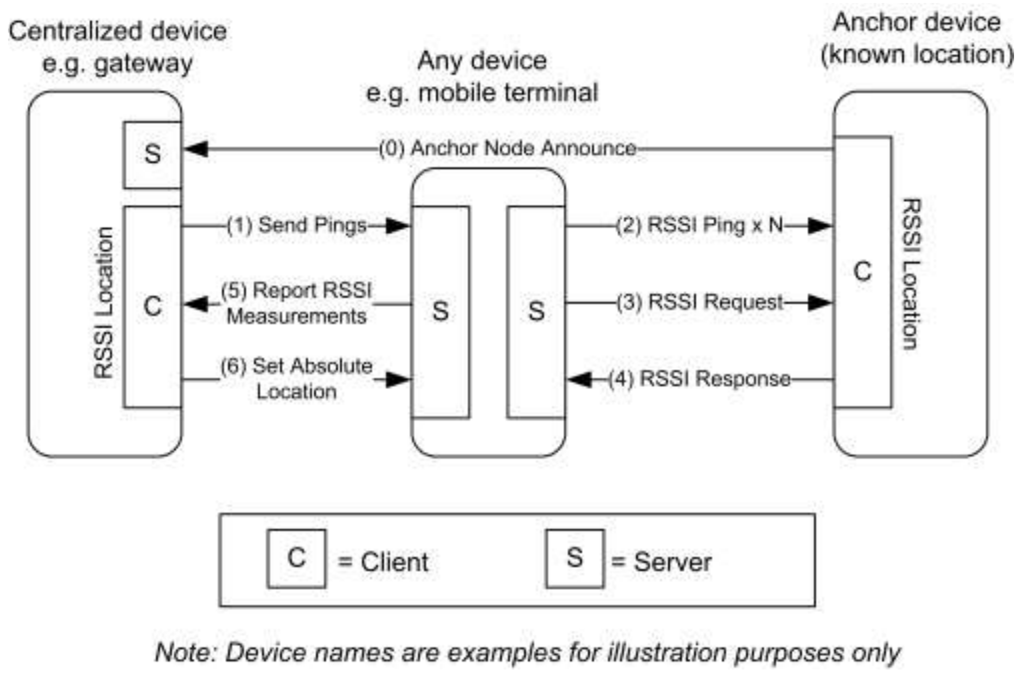

Network management device
Any device
C
Alarms
Configuration
Notification
S
C = Client
S = Server
Note: Device names are examples for illustration purposes only

3360 **Table 3-5. Other Clusters**

ID	Name	Description
0x000a	Time	Attributes and commands that provide an interface to a real-time clock.
0x000b	RSSI Location	Attributes and commands for exchanging location information and channel parameters among devices, and (optionally) reporting data to a centralized device that collects data from devices in the network and calculates their positions from the set of collected data.
0x0b05	Diagnostics	Attributes and commands that provide an interface to diagnostics of the stack
0x0020	Poll Control	Attributes and commands that provide an interface to control the polling of sleeping end device
0x001a	Power Profile	Attributes and commands that provide an interface to the power profile of a device
0x0b01	Meter Identification	Attributes and commands that provide an interface to meter identification

3361

3362 **Figure 3-4. Typical Usage of the Location Cluster with Centralized Device**



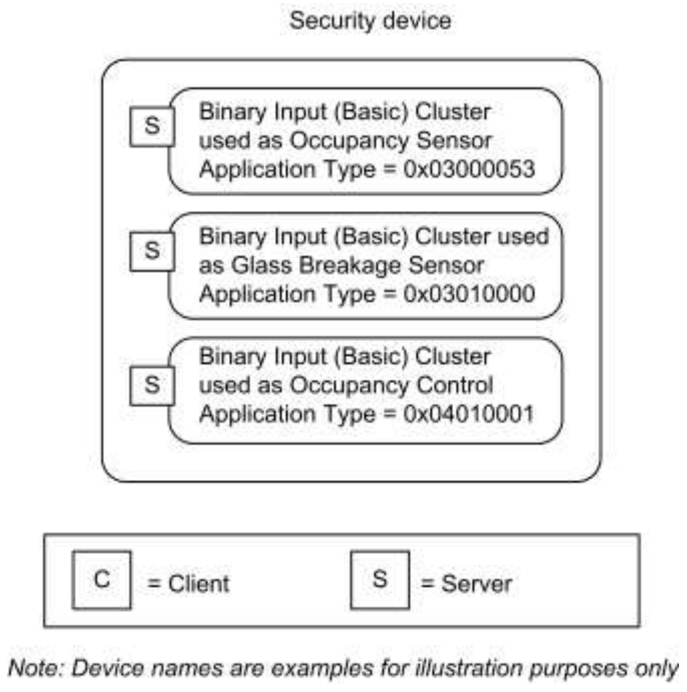
3363

3364 **Table 3-6. Generic Clusters**

ID	Cluster Name	Description
0x000c	Analog Input (basic)	An interface for reading the value of an analog measurement and accessing various characteristics of that measurement.

ID	Cluster Name	Description
0x000d	Analog Output (basic)	An interface for setting the value of an analog output (typically to the environment) and accessing various characteristics of that value.
0x000e	Analog Value (basic)	An interface for setting an analog value, typically used as a control system parameter, and accessing various characteristics of that value.
0x000f	Binary Input (basic)	An interface for reading the value of a binary measurement and accessing various characteristics of that measurement.
0x0010	Binary Output (basic)	An interface for setting the value of a binary output (typically to the environment) and accessing various characteristics of that value.
0x0011	Binary Value (basic)	An interface for setting a binary value, typically used as a control system parameter, and accessing various characteristics of that value.
0x0012	Multistate Input (basic)	An interface for reading the value of a multistate measurement and accessing various characteristics of that measurement.
0x0013	Multistate Output (basic)	An interface for setting the value of a multistate output (typically to the environment) and accessing various characteristics of that value.
0x0014	Multistate Value (basic)	An interface for setting a multistate value, typically used as a control system parameter, and accessing various characteristics of that value.

Figure 3-5. Example Usage of the Input, Output and Value Clusters


Security device
S
Binary Input (Basic) Cluster used as Occupancy Sensor Application Type = 0x03000053
S
Binary Input (Basic) Cluster used as Glass Breakage Sensor Application Type = 0x03010000
S
Binary Input (Basic) Cluster used as Occupancy Control Application Type = 0x04010001
C = Client
S = Server
Note: Device names are examples for illustration purposes only

3.2 Basic

3.2.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster supports an interface to the node or physical device. It provides attributes and commands for determining basic information, setting user information such as location, and resetting to factory defaults.

Note: Where a node supports multiple endpoints, it will often be the case that many of these settings will apply to the whole node, that is, they are the same for every endpoint on the node. In such cases, they can be implemented once for the node, and mapped to each endpoint.

3.2.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added; ZCLVersion set to 0x02
2	new attributes for manufacturer identification; CCB 1499 1584 2197 2229 ZCLVersion set to 0x03; ZLO 1.0

3.2.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	B

3.2.1.3 Cluster Identifiers

Identifier	Name
0x0000	Basic

3.2.2 Server

3.2.2.1 Dependencies

For the alarms functionality of this cluster to be operational, the Alarms cluster server SHALL be implemented on the same endpoint.

3.2.2.2 Attributes

The Basic cluster attributes are summarized in Table 3-7.

Table 3-7. Attributes of the Basic Cluster

Id	Name	Data Type	Range	Acc	Default	M/O
0x0000	*ZCLVersion*	uint8	0x00 to 0xff	R	0x03	M

Id	Name	Data Type	Range	Acc	Default	M/O
0x0001	*ApplicationVersion*	uint8	0x00 to 0xff	R	0x00	O
0x0002	*StackVersion*	uint8	0x00 to 0xff	R	0x00	O
0x0003	*HWVersion*	uint8	0x00 to 0xff	R	0x00	O
0x0004	*ManufacturerName*	string	0 to 32 bytes	R	empty string	O
0x0005	*ModelIdentifier*	string	0 to 32 bytes	R	empty string	O
0x0006	*DateCode*	string	0 to 16 bytes	R	empty string	O
0x0007	*PowerSource*	enum8	0x00 to 0xff	R	0x00	M
0x0008	*GenericDevice-Class*[8]	enum8	0x00 to 0xff	R	0xff	O
0x0009	*GenericDevice-Type*	enum8	0x00 to 0xff	R	0xff	O
0x000a	*ProductCode*	octstr		R	empty string	O
0x000b	*ProductURL*	string		R	empty string	O
0x000c	*ManufacturerVersionDetails*	string		R	empty string	O
0x000d	*SerialNumber*	string		R	empty string	O
0x000e	*ProductLabel*	string		R	empty string	O
0x0010	*LocationDescription*	string	0 to 16 bytes	RW	empty string	O
0x0011	*PhysicalEnvironment*	enum8	0x00 to 0xff	RW	0x00	O
0x0012	*DeviceEnabled*	bool	0x00 to 0x01	RW	0x01	O
0x0013	*AlarmMask*	map8	000000xx	RW	0x00	O
0x0014	*DisableLocalConfig*	map8	000000xx	RW	0x00	O
0x4000	*SWBuildID*	string	0 to 16 bytes	R	empty string	O

3.2.2.2.1 *ZCLVersion* Attribute

The *ZCLVersion* attribute represents a published set of foundation items (in Chapter 2), such as global commands and functional descriptions. **For this version of the ZCL, this attribute SHALL be set to 0x03.**

3.2.2.2.2 *ApplicationVersion* Attribute

The *ApplicationVersion* attribute is 8 bits in length and specifies the version number of the application software contained in the device. The usage of this attribute is manufacturer dependent.

3.2.2.2.3 *StackVersion* Attribute

The *StackVersion* attribute is 8 bits in length and specifies the version number of the implementation of the stack contained in the device. The usage of this attribute is manufacturer dependent.

[8] ZLO 1.0

3.2.2.2.4 *HWVersion* Attribute

The *HWVersion* attribute is 8 bits in length and specifies the version number of the hardware of the device. The usage of this attribute is manufacturer dependent.

3.2.2.2.5 *ManufacturerName* Attribute

The *ManufacturerName* attribute is a maximum of 32 bytes in length and specifies the name of the manufacturer as a character string.

3.2.2.2.6 *ModelIdentifier* Attribute

The *ModelIdentifier* attribute is a maximum of 32 bytes in length and specifies the model number (or other identifier) assigned by the manufacturer as a character string.

3.2.2.2.7 *DateCode* Attribute

The *DateCode* attribute is a character string with a maximum length of 16 bytes. The first 8 characters specify the date of manufacturer of the device in international date notation according to ISO 8601, i.e., YYYYMMDD, e.g., 20060814.

The final 8 characters MAY include country, factory, line, shift or other related information at the option of the manufacturer. The format of this information is manufacturer dependent.

3.2.2.2.8 PowerSource Attribute

The *PowerSource* attribute is 8 bits in length and specifies the source(s) of power available to the device. Bits b_0–b_6 of this attribute represent the primary power source of the device and bit b_7 indicates whether the device has a secondary power source in the form of a battery backup.

This attribute SHALL be set to one of the non-reserved values listed in Table 3-8. Bit 7 of this attribute SHALL be set to 1 if the device has a secondary power source in the form of a battery backup. Otherwise, bit 7 SHALL be set to 0.

Table 3-8. Values of the *PowerSource* Attribute

Value	Description
0x00	Unknown
0x01	Mains (single phase)
0x02	Mains (3 phase)
0x03	Battery
0x04	DC source
0x05	Emergency mains constantly powered
0x06	Emergency mains and transfer switch
Bit 7 set denotes battery backup source[9]	
0x80	Unknown

[9] CCB 2229 add enum values to represent the ‘mapped’ bit that means battery backup

Value	Description
0x81	Mains (single phase)
0x82	Mains (3 phase)
0x83	Battery
0x84	DC source
0x85	Emergency mains constantly powered
0x86	Emergency mains and transfer switch

3.2.2.2.9 GenericDeviceClass Attribute

The *GenericDeviceClass* attribute defines the field of application of the *GenericDeviceType* attribute. It SHALL be set to one of the non-reserved values listed below:

Table 3-9. Values of the *GenericDeviceClass* attribute

***GenericDeviceClass* value**	**Description**
0x00	Lighting
0x01 to 0xff	Reserved

3.2.2.2.10 GenericDeviceType Attribute

The *GenericDeviceType* attribute allows an application to show an icon on a rich user interface (e.g. smartphone app).

Notes on the usage of the *GenericDeviceType* attribute:

- lamps with integrated radio module SHALL have a proper value indicating the lamp type, according to the table below;
- devices that cannot be assigned to a proper category SHALL be set as "unspecified";

When the *GenericDeviceClass* attribute is set to 0x00 (i.e. lighting) the *GenericDeviceType* attribute SHALL be set to one of the non-reserved values listed below:

Table 3-10. Values of the *GenericDeviceType* attribute for the lighting class

Value	Description
0x00	Incandescent
0x01	Spotlight Halogen
0x02	Halogen bulb
0x03	CFL
0x04	Linear Fluorescent
0x05	LED bulb

Value	Description
0x06	Spotlight LED
0x07	LED strip
0x08	LED tube
0x09	Generic indoor luminaire/light fixture
0x0a	Generic outdoor luminaire/light fixture
0x0b	Pendant luminaire/light fixture
0x0c	Floor standing luminaire/light fixture
0xe0	Generic Controller (e.g. Remote controller)
0xe1	Wall Switch
0xe2	Portable remote controller
0xe3	Motion sensor / light sensor
0xe4 to 0xef	Reserved
0xf0	Generic actuator
0xf1	Wall socket
0xf2	Gateway/Bridge
0xf3	Plug-in unit
0xf4	Retrofit actuator
0xff	Unspecified

3.2.2.2.11 ProductCode Attribute

The *ProductCode* attribute allows an application to specify a code for the product. The *ProductCode* attribute SHALL have the format defined in Figure .

Octets:1	1	Variable
Octet Count	CodeId (see Table)	The code represented as a sequence of ASCII characters
	Octet data	

Figure 3-6. Format of the *ProductCode* attribute

Table 3-11. Values of the *CodeId* field of the *ProductCode* attribute

Code ID	Code type
0x00	Manufacturer defined
0x01	International article number (EAN)
0x02	Global trade item number (GTIN)
0x03	Universal product code (UPC)
0x04	Stock keeping unit (SKU)

In case no code has been provided, the Octet Count field SHALL be set to 0 (i.e. the octet string is empty).

3.2.2.2.12 ProductURL Attribute

The *ProductURL* attribute specifies a link to a web page containing specific product information.

Notes on the usage of the *ProductURL* attribute:

- The length of the URL SHALL be limited by the maximum number of bytes that can be transmitted from the application in a single frame. In most cases, such limit is around 50 bytes.
- In case no URL has been provided, the string SHALL be empty (i.e. the first byte is set to zero).

3.2.2.2.13 Manufacturer VersionDetails Attribute

Vendor specific human readable (displayable) string representing the versions of one of more program images supported on the device.

3.2.2.2.14 SerialNumber Attribute

Vendor specific human readable (displayable) serial number.

3.2.2.2.15 ProductLabel Attribute

Vendor specific human readable (displayable) product label.

3.2.2.2.16 *LocationDescription* Attribute

The *LocationDescription* attribute is a maximum of 16 bytes in length and describes the physical location of the device as a character string. This location description MAY be added into the device during commissioning.

3.2.2.2.17 *PhysicalEnvironment* Attribute

The *PhysicalEnvironment* attribute is 8 bits in length and specifies the type of physical environment in which the device will operate. This attribute SHALL be set to one of the non-reserved values listed in Table 3-12. All values are valid for endpoints supporting all profiles except when noted.

3461 **Table 3-12. Values of the *PhysicalEnvironment* Attribute**

Value	Description
0x00	Unspecified environment
0x01	Mirror Capacity Available – for 0x0109 Profile Id only; use 0x71 moving forward Atrium – defined for legacy devices with non-0x0109 Profile Id; use 0x70 moving forward Note: This value is deprecated for Profile Id 0x0104. The value 0x01 is maintained for historical purposes and SHOULD only be used for backwards compatibility with devices developed before this specification. The 0x01 value MUST be interpreted using the Profile Id of the endpoint upon which it is implemented. For endpoints with the Smart Energy Profile Id (0x0109) the value 0x01 has a meaning of Mirror. For endpoints with any other profile identifier, the value 0x01 has a meaning of Atrium.
0x02	Bar
0x03	Courtyard
0x04	Bathroom
0x05	Bedroom
0x06	Billiard Room
0x07	Utility Room
0x08	Cellar
0x09	Storage Closet
0x0a	Theater
0x0b	Office
0x0c	Deck
0x0d	Den
0x0e	Dining Room
0x0f	Electrical Room
0x10	Elevator
0x11	Entry
0x12	Family Room
0x13	Main Floor
0x14	Upstairs
0x15	Downstairs
0x16	Basement/Lower Level
0x17	Gallery
0x18	Game Room
0x19	Garage
0x1a	Gym
0x1b	Hallway

Value	Description
0x1c	House
0x1d	Kitchen
0x1e	Laundry Room
0x1f	Library
0x20	Master Bedroom
0x21	Mud Room (small room for coats and boots)
0x22	Nursery
0x23	Pantry
0x24	Office
0x25	Outside
0x26	Pool
0x27	Porch
0x28	Sewing Room
0x29	Sitting Room
0x2a	Stairway
0x2b	Yard
0x2c	Attic
0x2d	Hot Tub
0x2e	Living Room
0x2f	Sauna
0x30	Shop/Workshop
0x31	Guest Bedroom
0x32	Guest Bath
0x33	Powder Room (1/2 bath)
0x34	Back Yard
0x35	Front Yard
0x36	Patio
0x37	Driveway
0x38	Sun Room
0x39	Living Room
0x3a	Spa
0x3b	Whirlpool
0x3c	Shed

Value	Description
0x3d	Equipment Storage
0x3e	Hobby/Craft Room
0x3f	Fountain
0x40	Pond
0x41	Reception Room
0x42	Breakfast Room
0x43	Nook
0x44	Garden
0x45	Balcony
0x46	Panic Room
0x47	Terrace
0x48	Roof
0x49	Toilet
0x4a	Toilet Main
0x4b	Outside Toilet
0x4c	Shower room
0x4d	Study
0x4e	Front Garden
0x4f	Back Garden
0x50	Kettle
0x51	Television
0x52	Stove
0x53	Microwave
0x54	Toaster
0x55	Vacuum
0x56	Appliance
0x57	Front Door
0x58	Back Door
0x59	Fridge Door
0x60	Medication Cabinet Door
0x61	Wardrobe Door
0x62	Front Cupboard Door
0x63	Other Door

Value	Description
0x64	Waiting Room
0x65	Triage Room
0x66	Doctor’s Office
0x67	Patient’s Private Room
0x68	Consultation Room
0x69	Nurse Station
0x6a	Ward
0x6b	Corridor
0x6c	Operating Theatre
0x6d	Dental Surgery Room
0x6e	Medical Imaging Room
0x6f	Decontamination Room
0x70	Atrium[10]
0x71	Mirror
0xff	Unknown environment

3.2.2.2.18 DeviceEnabled Attribute

The *DeviceEnabled* attribute is a Boolean and specifies whether the device is enabled or disabled. This attribute SHALL be set to one of the non-reserved values listed in Table 3-13.

Table 3-13. Values of the *DeviceEnable* Attribute

***DeviceEnable* Attribute Value**	**Description**
0x00	Disabled
0x01	Enabled

'Disabled' means that the device does not send or respond to application level commands, other than commands to read or write attributes. Values of attributes which depend on the operation of the application MAY be invalid, and any functionality triggered by writing to such attributes MAY be disabled. Networking functionality remains operational.

If implemented, the identify cluster cannot be disabled, i.e., it remains functional regardless of this setting.

3.2.2.2.19 AlarmMask Attribute

The *AlarmMask* attribute is 8 bits in length and specifies which of a number of general alarms MAY be generated, as listed in Table 3-14. A ‘1’ in each bit position enables the associated alarm.

[10] CCB 2197 new values for Atrium and Mirror because of conflict with ZSE for value 0x01

Table 3-14. Values of the *AlarmMask* Attribute

Attribute Bit Number	Alarm Code	Alarm
0	0	General hardware fault
1	1	General software fault

These alarms are provided as basic alarms that a device MAY use even if no other clusters with alarms are present on the device.

3.2.2.2.20 DisableLocalConfig Attribute

The *DisableLocalConfig* attribute allows a number of local device configuration functions to be disabled.

Table 3-15. Values of the *DisableLocalConfig* Attribute

Attribute Bit Number	Description
0	0 = Reset (to factory defaults) enabled 1 = Reset (to factory defaults) disabled
1	0 = Device configuration enabled 1 = Device configuration disabled

The intention of this attribute is to allow disabling of any local configuration user interface, for example to prevent reset or binding buttons being activated by non-authorized persons in a public building.

Bit 0 of the *DisableLocalConfig* attribute disables any factory reset button (or equivalent) on the device. Bit 1 disables any device configuration button(s) (or equivalent)—for example, a bind button.

3.2.2.2.21 SWBuildID Attribute

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

3.2.2.3 Commands Received

The command IDs for the Basic cluster are listed in Table 3-16.

Table 3-16. Received Command IDs for the Basic Cluster

Command Identifier	Description	M/O
0x00	Reset to Factory Defaults	O

3.2.2.3.1 Reset to Factory Defaults Command

This command does not have a payload.

3.2.2.3.1.1 Effect on Receipt

On receipt of this command, the device resets all the attributes of all its clusters to their factory defaults.

Note that networking functionality, bindings, groups, or other persistent data are not affected by this command.

3.2.2.4 Commands Generated

No commands are generated by the server cluster.

3.2.3 Client

The client has no dependencies or attributes. No cluster specific commands are received by the client.

The cluster specific commands generated by the client cluster are those received by the server, as required by the application.

3.3 Power Configuration

3.3.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

Attributes for determining detailed information about a device’s power source(s) and for configuring under/over voltage alarms.

3.3.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added; mains power lost alarm added to *MainsAlarmMask*; CCB 1809

3.3.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	PC

3.3.1.3 Cluster Identifiers

Identifier	Name
0x0001	Power Configuration

3.3.2 Server

3.3.2.1 Dependencies

Any endpoint that implements this server cluster SHALL also implement the Basic server cluster.

For the alarm functionality described in this cluster to be operational, any endpoint that implements the Power Configuration server cluster must also implement the Alarms server cluster (see sub-clause Alarms).

3.3.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 3-17.

Table 3-17. Power Configuration Attribute Sets

Attribute Set Identifier	Description
0x000	Mains Information
0x001	Mains Settings
0x002	Battery Information
0x003	Battery Settings
0x004	Battery Source 2 Information
0x005	Battery Source 2 Settings
0x006	Battery Source 3 Information
0x007	Battery Source 3 Settings

3.3.2.2.1 Mains Information Attribute Set

The Mains Information attribute set contains the attributes summarized in Table 3-18.

Table 3-18. Attributes of the Mains Information Attribute Set

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*MainsVoltage*	uint16	0x0000 to 0xffff	R	-	O
0x0001	*MainsFrequency*	uint8	0x00 to 0xff	R	-	O

3.3.2.2.1.1 *MainsVoltage* Attribute

The *MainsVoltage* attribute is 16 bits in length and specifies the actual (measured) RMS voltage (or DC voltage in the case of a DC supply) currently applied to the device, measured in units of 100mV.

3.3.2.2.1.2 MainsFrequency Attribute

The *MainsFrequency* attribute is 8 bits in length and represents the frequency, in Hertz, of the mains as determined by the device as follows:

MainsFrequency = 0.5 x measured frequency

Where 2 Hz <= measured frequency <= 506 Hz, corresponding to a *MainsFrequency* in the range 1 to 0xfd.

The maximum resolution this format allows is 2 Hz.

The following special values of *MainsFrequency* apply.

0x00 indicates a frequency that is too low to be measured.

0xfe indicates a frequency that is too high to be measured.

0xff indicates that the frequency could not be measured.

In the case of a DC supply, this attribute SHALL also have the value zero.

3.3.2.2.2 Mains Settings Attribute Set

The Mains Settings attribute set contains the attributes summarized in Table 3-19.

Table 3-19. Attributes of the Mains Settings Attribute Set

Identifier	Name	Type	Range	Access	Default	M/O
0x0010	*MainsAlarmMask*	map8	0000 00xx	RW	0000 0000	O
0x0011	*MainsVoltageMinThreshold*	uint16	0x0000 to 0xffff	RW	0x0000	O
0x0012	*MainsVoltageMaxThreshold*	uint16	0x0000 to 0xffff	RW	0xffff	O
0x0013	*MainsVoltageDwellTripPoint*	uint16	0x0000 to 0xffff	RW	0x0000	O

The alarm settings in this table require the Alarms cluster to be implemented on the same device - see Dependencies. If the Alarms cluster is not present on the same device, they MAY be omitted.

3.3.2.2.2.1 *MainsAlarmMask* Attribute

The *MainsAlarmMask* attribute is 8 bits in length and specifies which mains alarms MAY be generated, as listed in Table 3-20. A ‘1’ in each bit position enables the alarm.

Table 3-20. Values of the *MainsAlarmMask* Attribute

MainsAlarmMask Attribute Bit Number	Alarm	Rev
0	Mains Voltage too low (3.3.2.2.2.2)	0
1	Mains Voltage too high (3.3.2.2.2.3)	0
2	Mains power supply lost/unavailable (i.e., device is running on battery)	1

3.3.2.2.2.2 MainsVoltageMinThreshold Attribute

The *MainsVoltageMinThreshold* attribute is 16 bits in length and specifies the lower alarm threshold, measured in units of 100mV, for the MainsVoltage attribute. The value of this attribute SHALL be less than MainsVoltageMaxThreshold.

If the value of MainsVoltage drops below the threshold specified by MainsVoltageMinThreshold, the device SHALL start a timer to expire after MainsVoltageDwellTripPoint seconds. If the value of this attribute increases to greater than or equal to MainsVoltageMinThreshold before the timer expires, the device SHALL stop and reset the timer. If the timer expires, an alarm SHALL be generated.

The Alarm Code field (see 3.11.2.4.1) included in the generated alarm SHALL be 0x00.

If this attribute takes the value 0xffff then this alarm SHALL NOT be generated.

3.3.2.2.2.3 MainsVoltageMaxThreshold Attribute

The *MainsVoltageMaxThreshold* attribute is 16 bits in length and specifies the upper alarm threshold, measured in units of 100mV, for the *MainsVoltage* attribute. The value of this attribute SHALL be greater than *MainsVoltageMinThreshold.*

If the value of *MainsVoltage* rises above the threshold specified by *MainsVoltageMaxThreshold*, the device SHALL start a timer to expire after *MainsVoltageDwellTripPoint* seconds. If the value of this attribute drops to lower than or equal to *MainsVoltageMaxThreshold* before the timer expires, the device SHALL stop and reset the timer. If the timer expires, an alarm SHALL be generated.

The Alarm Code field (see 3.11.2.4.1) included in the generated alarm SHALL be 0x01.

If this attribute takes the value 0xffff then this alarm SHALL NOT be generated.

3.3.2.2.2.4 MainsVoltageDwellTripPoint Attribute

The *MainsVoltageDwellTripPoint* attribute is 16 bits in length and specifies the length of time, in seconds that the value of *MainsVoltage* MAY exist beyond either of its thresholds before an alarm is generated.

If this attribute takes the value 0xffff then the associated alarms SHALL NOT be generated.

3.3.2.2.3 Battery Information Attribute Set

The Battery Information attribute set contains the attributes summarized in Table 3-21.

Table 3-21. Attributes of the Battery Information Attribute Set

Identifier	Name	Type	Range	Access	Default	M/O
0x0020	*BatteryVoltage*	uint8	0x00 to 0xff	R	-	O
0x0021	*BatteryPercentageRemaining*	uint8	0x00 to 0xff	R Reportable	0	O

Manufacturers SHOULD measure the battery voltage and capacity at a consistent moment (e.g., the moment of radio transmission (i.e., peak demand)) in order to avoid unnecessary fluctuation in reporting the attribute, which can confuse users and make them call into question the quality of the device.

Manufacturers SHOULD employ a hysteresis algorithm appropriate for their battery type in order to smooth battery reading fluctuations and avoid sending multiple battery warning messages when crossing the voltage thresholds defined for warnings.

3.3.2.2.3.1 BatteryVoltage Attribute

The *BatteryVoltage* attribute is 8 bits in length and specifies the current actual (measured) battery voltage, in units of 100mV.

The value 0xff indicates an invalid or unknown reading.

3.3.2.2.3.2 BatteryPercentageRemaining Attribute

Specifies the remaining battery life as a half integer percentage of the full battery capacity (e.g., 34.5%, 45%, 68.5%, 90%) with a range between zero and 100%, with 0x00 = 0%, 0x64 = 50%, and 0xC8 = 100%. This is particularly suited for devices with rechargeable batteries.

The value 0xff indicates an invalid or unknown reading.

This attribute SHALL be configurable for attribute reporting.

3.3.2.2.4 Battery Settings Attribute Set

Table 3-22. Attributes of the Battery Settings Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0030	*BatteryManufacturer*	string	0 to 16 bytes	RW	Empty string	O
0x0031	*BatterySize*	enum8	0x00 to 0xff	RW	0xff	O
0x0032	*BatteryAHrRating*	uint16	0x0000 to 0xffff	RW	-	O
0x0033	*BatteryQuantity*	uint8	0x00 to 0xff	RW	-	O
0x0034	*BatteryRatedVoltage*	uint8	0x00 to 0xff	RW	-	O
0x0035	*BatteryAlarmMask*	map8	0000 000x	RW	0000 0000	O
0x0036	*BatteryVoltageMinThreshold*	uint8	0x00 to 0xff	RW	0x0000	O
0x0037	*BatteryVoltageThreshold1*	uint8	0x00 to 0xff	R*W	0x00	O
0x0038	*BatteryVoltageThreshold2*	uint8	0x00 to 0xff	R*W	0x00	O
0x0039	*BatteryVoltageThreshold3*	uint8	0x00 to 0xff	R*W	0x00	O
0x003a	*BatteryPercentageMinThreshold*	uint8	0x00 to 0xff	R*W	0x00	O
0x003b	*BatteryPercentageThreshold1*	uint8	0x00 to 0xff	R*W	0x00	O
0x003c	*BatteryPercentageThreshold2*	uint8	0x00 to 0xff	R*W	0x00	O
0x003d	*BatteryPercentageThreshold3*	uint8	0x00 to 0xff	R*W	0x00	O
0x003e	*BatteryAlarmState*	map32	0x00…x	Read	0x000…0	O

3.3.2.2.4.1 BatteryManufacturer Attribute

The *BatteryManufacturer* attribute is a maximum of 16 bytes in length and specifies the name of the battery manufacturer as a character string.

3.3.2.2.4.2 BatterySize Attribute

The *BatterySize* attribute is an enumeration which specifies the type of battery being used by the device. This attribute SHALL be set to one of the non-reserved values listed in Table 3-23.

Table 3-23. Values of the *BatterySize* Attribute

Attribute Value	Description
0x00	No battery
0x01	Built in
0x02	Other

Attribute Value	Description
0x03	AA
0x04	AAA
0x05	C
0x06	D
0x07	CR2 (IEC: CR17355 / ANSI: 5046LC)
0x08	CR123A (IEC: CR17345 / ANSI: 5018LC)
0xff	Unknown

3.3.2.2.4.3 *BatteryAHrRating* Attribute

The *BatteryAHrRating* attribute is 16 bits in length and specifies the Ampere-hour rating of the battery, measured in units of 10mAHr.

3.3.2.2.4.4 BatteryQuantity Attribute

The *BatteryQuantity* attribute is 8 bits in length and specifies the number of battery cells used to power the device.

3.3.2.2.4.5 BatteryRatedVoltage Attribute

The *BatteryRatedVoltage* attribute is 8 bits in length and specifies the rated voltage of the battery being used in the device, measured in units of 100mV.

3.3.2.2.4.6 BatteryAlarmMask Attribute

The *BatteryAlarmMask* attribute specifies which battery alarms must be generated, as listed in Table 3-24. A '1' in each bit position enables the alarm.

Table 3-24. Values of the *BatteryAlarmMask* Attribute

BatteryAlarmMask* Attribute Bit Number	**Description**
0	Battery voltage too low to continue operating the device's radio (i.e., BatteryVoltageMinThreshold value has been reached)
1	Battery Alarm 1 (i.e., Battery Voltage Threshold 1 or Battery Percentage Threshold 1 value has been reached)
2	Battery Alarm 2 (i.e., Battery Voltage Threshold 2 or Battery Percentage Threshold 2 value has been reached)
3	Battery Alarm 3 (i.e., Battery Voltage Threshold 3 or Battery Percentage Threshold 3 value has been reached)

Manufacturers are responsible for determining the capability to sense and levels at which the alarms are generated. See Section 10.3.2, References, for additional recommendations on measuring battery voltage.

3.3.2.2.4.7 *BatteryVoltageMinThreshold* Attribute

Specifies the low battery voltage alarm threshold, measured in units of 100mV at which the device can no longer operate or transmit via its radio (i.e., last gasp).

If the value of *BatteryVoltage* drops below the threshold specified by *BatteryVoltageMinThreshold*, an appropriate alarm SHALL be generated and/or the corresponding bit SHALL be updated in the *BatteryAlarmState* attribute.

In order to report to Power Configuration clients, servers that implement *BatteryVoltageMinThreshold* attribute SHALL implement alarming via the Alarm Cluster, attribute reporting via the *BatteryAlarmState* attribute, or both.

For servers that implement alarming via the Alarm Cluster, the appropriate alarm is specified in the Alarm Code field (see 3.11.2.3.1) included in the generated alarm and SHALL be one of the values in Table 3-25. The host determines which alarm code to populate based on the *BatteryAlarmMask* attribute and the *BatteryVoltageMinThreshold* attribute reached. For example, when the *BatteryVoltage* attribute reaches the value specified by the *BatteryVoltageMinThreshold* attribute, an alarm with the Alarm Code Field Enumeration "0x10" SHALL be generated.

For servers that implement battery alarm reporting via the *BatteryAlarmState* attribute, the bit corresponding to the threshold level reached SHALL be set to TRUE. See the *BatteryAlarmState* attribute details for more information.

Table 3-25. Alarm Code Field Enumerations for Battery Alarms

Enum	Description
0x10	*BatteryVoltageMinThreshold* or *BatteryPercentageMinThreshold* reached for Battery Source 1
0x11	*BatteryVoltageThreshold1* or *BatteryPercentageThreshold1* reached for Battery Source 1
0x12	*BatteryVoltageThreshold2* or *BatteryPercentageThreshold2* reached for Battery Source 1
0x13	*BatteryVoltageThreshold3* or *BatteryPercentageThreshold3* reached for Battery Source 1
0x20	*BatteryVoltageMinThreshold* or *BatteryPercentageMinThreshold* reached for Battery Source 2
0x21	*BatteryVoltageThreshold1* or *BatteryPercentageThreshold1* reached for Battery Source 2
0x22	*BatteryVoltageThreshold2* or *BatteryPercentageThreshold2* reached Battery Source 2
0x23	*BatteryVoltageThreshold3* or *BatteryPercentageThreshold3* reached Battery Source 2
0x30	*BatteryVoltageMinThreshold* or *BatteryPercentageMinThreshold* reached for Battery Source 3
0x31	*BatteryVoltageThreshold1* or *BatteryPercentageThreshold1* reached for Battery Source 3
0x32	*BatteryVoltageThreshold2* or *BatteryPercentageThreshold2* reached Battery Source 3
0x33	*BatteryVoltageThreshold3* or *BatteryPercentageThreshold3* reached Battery Source 3
0x3a	Mains power supply lost/unavailable (i.e., device is running on battery)
0xff	Alarm SHALL NOT be generated

3.3.2.2.4.8 BatteryVoltageThreshold 1-3 Attributes

Specify the low voltage alarm thresholds, measured in units of 100mV, for the *BatteryVoltage* attribute.

If the value of *BatteryVoltage* drops below the threshold specified by a *BatteryVoltageThreshold*, an appropriate alarm SHALL be generated and/or the corresponding bit SHALL be updated in the *BatteryAlarmState* attribute.

The *BatteryVoltageThreshold1-3* attributes SHALL be ordered in ascending order such that the *BatteryVoltage* level specified to trigger:

- *BatteryVoltageThreshold3* is higher than the level specified to trigger *BatteryVoltageThreshold2*
- *BatteryVoltageThreshold2* is higher than the level specified to trigger *BatteryVoltageThreshold*
- *BatteryVoltageThreshold1* is higher than the level specified to trigger *BatteryVoltageMinThreshold*

The appropriate alarm is specified in the Alarm Code field (see 3.11.2.3.1) included in the generated alarm and SHALL be one of the values in Table 3-25. The host determines which alarm code to populate based on the *BatteryAlarmMask* attribute and the *BatteryVoltageThreshold1-3* attribute reached.

If this attribute takes the value 0xff then this alarm SHALL NOT be generated.

3.3.2.2.4.9 BatteryPercentageMinThreshold Attribute

Specifies the low battery percentage alarm threshold, measured in percentage (i.e., zero to 100%), for the *BatteryPercentageRemaining* attribute (see sub-clause 3.3.2.2.3.2).

If the value of *BatteryPercentageRemaining* drops below the threshold specified by a *BatteryPercentageThreshold*, an appropriate alarm SHALL be generated.

The appropriate alarm is specified in the Alarm Code field (see 3.11.2.3.1) included in the generated alarm and SHALL be the value in Table 3-25 that corresponds with this threshold being reached for a given battery source. The host determines which alarm code to populate based on the *BatteryAlarmMask* attribute.

If this attribute takes the value 0xff then this alarm SHALL NOT be generated.

3.3.2.2.4.10 BatteryPercentageThreshold 1-3 Attributes

Specify the low battery percentage alarm thresholds, measured in percentage (i.e., zero to 100%), for the *BatteryPercentageRemaining* attribute (see sub-clause 3.3.2.2.3.2).

If the value of *BatteryPercentageRemaining* drops below the threshold specified by a *BatteryPercentageThreshold*, an appropriate alarm SHALL be generated.

The *BatteryPercentageThreshold1-3* attributes SHALL be ordered in ascending order such that the *BatteryPercentageRemaining* level specified to trigger:

- BatteryPercentageThreshold3 is higher than the level specified to trigger BatteryPercentageThreshold2
- BatteryPercentageThreshold2 is higher than the level specified to trigger BatteryPercentageThreshold
- BatteryPercentageThreshold1 is higher than the level specified to trigger BatteryPercentageMinThreshold

The appropriate alarm is specified in the Alarm Code field (see 3.11.2.3.1) included in the generated alarm and SHALL be one of the values in Table 3-25. The host determines which alarm code to populate based on the *BatteryAlarmMask* attribute and the *BatteryPercentageThreshold1-3* attribute reached.

If this attribute takes the value 0xff then this alarm SHALL NOT be generated.

3.3.2.2.4.11 BatteryAlarmState Attribute

Specifies the current state of the device's battery alarms. This attribute provides a persistent record of a device's battery alarm conditions as well as a mechanism for reporting changes to those conditions, including the elimination of battery alarm states (e.g., when a battery is replaced).

If implemented, the server SHALL support attribute reporting for *BatteryAlarmState* attribute. This provides clients with a mechanism for reading the current state in case they missed the initial attribute report and also reduces network and battery use due to repeated polling of this attribute when it has not changed. It also provides a way of notifying clients when battery alarm conditions no longer exist (e.g., when the batteries have been replaced).

Table 3-26. *BatteryAlarmState* Enumerations

Bit	Description
0	*BatteryVoltageMinThreshold* or *BatteryPercentageMinThreshold* reached for Battery Source 1
1	*BatteryVoltageThreshold1* or *BatteryPercentageThreshold1* reached for Battery Source 1
2	*BatteryVoltageThreshold2* or *BatteryPercentageThreshold2* reached for Battery Source 1
3	*BatteryVoltageThreshold3* or *BatteryPercentageThreshold3* reached for Battery Source 1
10	*BatteryVoltageMinThreshold* or *BatteryPercentageMinThreshold* reached for Battery Source 2
11	*BatteryVoltageThreshold1* or *BatteryPercentageThreshold1* reached for Battery Source 2
12	*BatteryVoltageThreshold2* or *BatteryPercentageThreshold2* reached Battery Source 2
13	*BatteryVoltageThreshold3* or *BatteryPercentageThreshold3* reached Battery Source 2
20	*BatteryVoltageMinThreshold* or *BatteryPercentageMinThreshold* reached for Battery Source 3
21	*BatteryVoltageThreshold1* or *BatteryPercentageThreshold1* reached for Battery Source 3
22	*BatteryVoltageThreshold2* or *BatteryPercentageThreshold2* reached Battery Source 3
23	*BatteryVoltageThreshold3* or *BatteryPercentageThreshold3* reached Battery Source 3
30	Mains power supply lost/unavailable (i.e., device is running on battery)

Manufacturers are responsible for determining the capability to sense and levels at which the alarms are generated. See 3.3.2.2.3 for additional recommendations on measuring battery voltage.

3.3.2.2.5 Battery Information 2 Attribute Set

This attribute set is an exact replica of all the attributes, commands, and behaviors contained within the Battery Information Attribute Set and provides a host with the ability to represent battery information for a secondary battery bank or cell.

3.3.2.2.6 Battery Settings 2 Attribute Set

This attribute set is an exact replica of all the attributes, commands, and behaviors contained within the Battery Settings Attribute Set and provides a host with the ability to represent battery settings for a secondary battery bank or cell.

3.3.2.2.7 Battery Information 3 Attribute Set

This attribute set is an exact replica of all the attributes, commands, and behaviors contained within the Battery Information Attribute Set and provides a host with the ability to represent battery information for a tertiary battery bank or cell.

3.3.2.2.8 Battery Settings 3 Attribute Set

This attribute set is an exact replica of all the attributes, commands, and behaviors contained within the Battery Settings Attribute Set and provides a host with the ability to represent battery settings for a tertiary battery bank or cell. Commands Received

3.3.2.3 Commands Received

No commands are received by the server.

3.3.2.4 Commands Generated

The server generates no commands.

3.3.3 Client

The client has no dependencies or cluster specific attributes. There are no cluster specific commands that are generated or received by the client

3.4 Device Temperature Configuration

3.4.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

Attributes for determining information about a device's internal temperature, and for configuring under/over temperature alarms for temperatures that are outside the device's operating range.

3.4.1.1 Revision History

Rev	Description
1	global mandatory *Cluster Revision* attribute added

3.4.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	DTMP

3.4.1.3 Cluster Identifiers

Identifier	Name
0x0002	Device Temperature Configuration

3.4.2 Server

3.4.2.1 Dependencies

For the alarm functionality described in this cluster to be operational, any endpoint that implements the Device Temperature Configuration server cluster SHALL also implement the Alarms server cluster (see 3.11).

3.4.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 3-27.

Table 3-27. Device Temperature Configuration Attribute Sets

Attribute Set Identifier	Description
0x000	Device Temperature Information
0x001	Device Temperature Settings

3.4.2.2.1 Device Temperature Information Attribute Set

The Device Temperature Information attribute set contains the attributes summarized in Table 3-28.

Table 3-28. Device Temperature Information Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0000	*CurrentTemperature*	int16	-200 to +200	R	-	M
0x0001	*MinTempExperienced*	int16	-200 to +200	R	-	O
0x0002	*MaxTempExperienced*	int16	-200 to +200	R	-	O
0x0003	*OverTempTotalDwell*	uint16	0x0000 to 0xffff	R	0	O

3.4.2.2.1.1 CurrentTemperature Attribute

The *CurrentTemperature* attribute is 16 bits in length and specifies the current internal temperature, in degrees Celsius, of the device. This attribute SHALL be specified in the range –200 to +200.

The value 0xffff indicates an invalid reading.

3.4.2.2.1.2 MinTempExperienced Attribute

The *MinTempExperienced* attribute is 16 bits in length and specifies the minimum internal temperature, in degrees Celsius, the device has experienced while powered. This attribute SHALL be specified in the range –200 to +200.

The value 0xffff indicates an invalid reading.

3.4.2.2.1.3 MaxTempExperienced Attribute

The *MaxTempExperienced* attribute is 16 bits in length and specifies the maximum internal temperature, in degrees Celsius, the device has experienced while powered. This attribute SHALL be specified in the range –200 to +200.

The value 0xffff indicates an invalid reading.

3.4.2.2.1.4 OverTempTotalDwell Attribute

The *OverTempTotalDwell* attribute is 16 bits in length and specifies the length of time, in hours; the device has spent above the temperature specified by the *HighTempThreshold* attribute 3.4.2.2.2.3, cumulative over the lifetime of the device.

The value 0xffff indicates an invalid time.

3.4.2.2.2 Device Temperature Settings Attribute Set

The Device Temperature Settings attribute set contains the attributes summarized in Table 3-29.

Table 3-29. Device Temperature Settings Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0010	*DeviceTempAlarmMask*	map8	0000 00xx	RW	0000 0000	O
0x0011	*LowTempThreshold*	int16	-200 to +200	RW	-	O
0x0012	*HighTempThreshold*	int16	-200 to +200	RW	-	O
0x0013	*LowTempDwellTripPoint*	uint24	0x000000 to 0xffffff	RW	-	O
0x0014	*HighTempDwellTripPoint*	uint24	0x000000 to 0xffffff	RW	-	O

All attributes in this table require the Alarms cluster to be implemented on the same device - see Dependencies. If the Alarms cluster is not present on the same device, they MAY be omitted.

3.4.2.2.2.1 *DeviceTempAlarmMask* Attribute

The *DeviceTempAlarmMask* attribute is 8 bits in length and specifies which alarms MAY be generated, as listed in Table 3-30. A ‘1’ in each bit position enables the corresponding alarm.

Table 3-30. Values of the *DeviceTempAlarmMask* Attribute

Attribute Bit Number	Alarm
0	Device Temperature too low
1	Device Temperature too high

3.4.2.2.2.2 LowTempThreshold Attribute

The *LowTempThreshold* attribute is 16 bits in length and specifies the lower alarm threshold, measured in degrees Celsius (range -200°C to 200°C), for the *CurrentTemperature* attribute. The value of this attribute SHALL be less than *HighTempThreshold.*

If the value of *CurrentTemperature* drops below the threshold specified by *LowTempThreshold*, the device SHALL start a timer to expire after *LowTempDwellTripPoint* seconds. If the value of this attribute increases to greater than or equal to *LowTempThreshold* before the timer expires, the device SHALL stop and reset the timer. If the timer expires, an alarm SHALL be generated.

The Alarm Code field (see 3.11.2.4.1) included in the generated alarm SHALL be 0x00.

If this attribute takes the value 0x8000 then this alarm SHALL NOT be generated.

3.4.2.2.2.3 HighTempThreshold Attribute

The *HighTempThreshold* attribute is 16 bits in length and specifies the upper alarm threshold, measured in degrees Celsius (range -200°C to 200°C), for the *CurrentTemperature* attribute. The value of this attribute SHALL be greater than *LowTempThreshold.*

If the value of *CurrentTemperature* rises above the threshold specified by *HighTempThreshold*, the device SHALL start a timer to expire after *HighTempDwellTripPoint* seconds. If the value of this attribute drops to lower than or equal to *HighTempThreshold* before the timer expires, the device SHALL stop and reset the timer. If the timer expires, an alarm SHALL be generated.

The Alarm Code field (see 3.11.2.4.1) included in the generated alarm SHALL be 0x01.

If this attribute takes the value 0x8000 then this alarm SHALL NOT be generated.

3.4.2.2.2.4 LowTempDwellTripPoint Attribute

The *LowTempDwellTripPoint* attribute is 24 bits in length and specifies the length of time, in seconds, that the value of *CurrentTemperature* MAY exist below *LowTempThreshold* before an alarm is generated.

If this attribute takes the value 0xffffff then this alarm SHALL NOT be generated.

3.4.2.2.2.5 HighTempDwellTripPoint Attribute

The *HighTempDwellTripPoint* attribute is 24 bits in length and specifies the length of time, in seconds, that the value of *CurrentTemperature* MAY exist above HighTempThreshold before an alarm is generated.

If this attribute takes the value 0xffffff then this alarm SHALL NOT be generated.

3.4.2.3 Commands

No commands are received or generated by the server.

3.4.3 Client

The client has no dependencies or cluster specific attributes. There are no cluster specific commands that are generated or received by the client.

3.5 Identify

3.5.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

Attributes and commands to put a device into an Identification mode (e.g., flashing a light), that indicates to an observer – e.g., an installer - which of several devices it is, also to request any device that is identifying itself to respond to the initiator.

Note that this cluster cannot be disabled, and remains functional regardless of the setting of the *DeviceEnable* attribute in the Basic cluster.

3.5.1.1 Revision History

Rev	Description
1	global mandatory *Cluster Revision* attribute added

3.5.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	I

3.5.1.3 Cluster Identifiers

Identifier	Name
0x0003	Identify

3.5.2 Server

3.5.2.1 Attributes

The server supports the attribute shown in Table 3-31.

Table 3-31. Attributes of the Identify Server Cluster

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*IdentifyTime*	uint16	0x0000 to 0xffff	RW	0x0000	M

3.5.2.1.1 *IdentifyTime* Attribute

The *IdentifyTime* attribute specifies the remaining length of time, in seconds, that the device will continue to identify itself.

If this attribute is set to a value other than 0x0000 then the device SHALL enter its identification procedure, in order to indicate to an observer which of several devices it is. It is recommended that this procedure consists of flashing a light with a period of 0.5 seconds. The *IdentifyTime* attribute SHALL be decremented every second.

If this attribute reaches or is set to the value 0x0000 then the device SHALL terminate its identification procedure.

3.5.2.2 Commands Received

The server side of the identify cluster is capable of receiving the commands listed in Table 3-32.

Table 3-32. Received Command IDs for the Identify Cluster

Command Identifier	Description	M/O
0x00	Identify	M
0x01	Identify Query	M
0x40	Trigger effect	O

3.5.2.2.1 Identify Command

The identify command starts or stops the receiving device identifying itself.

3.5.2.2.1.1 Payload Format

The identify query response command payload SHALL be formatted as illustrated in Figure 3-7.

Figure 3-7. Format of Identify Query Response Command Payload

Octets	2
Data Type	uint16
Field Name	Identify Time

3.5.2.2.1.2 Effect on Receipt

On receipt of this command, the device SHALL set the *IdentifyTime* attribute to the value of the Identify Time field. This then starts, continues, or stops the device's identification procedure as detailed in 3.5.2.1.1.

3.5.2.2.2 Identify Query Command

The identify query command allows the sending device to request the target or targets to respond if they are currently identifying themselves.

This command has no payload.

3.5.2.2.2.1 Effect on Receipt

On receipt of this command, if the device is currently identifying itself then it SHALL generate an appropriate Identify Query Response command, see 3.5.2.4.1 and unicast it to the requestor. If the device is not currently identifying itself it SHALL take no further action.

3.5.2.2.3 Trigger Effect Command

The *Trigger Effect* command allows the support of feedback to the user, such as a certain light effect. It is used to allow an implementation to provide visual feedback to the user under certain circumstances such as a color light turning green when it has successfully connected to a network. The use of this command and the effects themselves are entirely up to the implementer to use whenever a visual feedback is useful but it is not the same as and does not replace the identify mechanism used during commissioning.

The payload of this command SHALL be formatted as illustrated in Figure 3-8.

Figure 3-8. Format of the Trigger Effect Command

Octets	1	1
Data Type	uint8	uint8
Field Name	Effect identifier	Effect variant

3.5.2.2.3.1 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 nonreserved values listed in Table 3-33.

Table 3-33. Values of the Effect Identifier Field of the Trigger Effect Command

Effect Identifier Field Value	**Effect**[11]	**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; noncolored light flashes twice.
0x0b	Channel change	e.g., Colored light turns orange for 8 seconds; noncolored 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.

3.5.2.2.3.2 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 3-34.

Table 3-34. Values of the Effect Variant Field of the Trigger Effect Command

Effect Variant Field Value	**Description**
0x00	Default

3.5.2.2.3.3 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.

[11] Implementers SHOULD indicate during testing how they handle each effect.

3.5.2.3 Commands Generated

The server side of the identify cluster is capable of generating the commands listed in Table 3-35.

Table 3-35. Generated Command IDs for the Identify Cluster

Command Identifier Field Value	Description	M/O
0x00	Identify Query Response	M

3.5.2.3.1 Identify Query Response Command

The identify query response command is generated in response to receiving an Identify Query command, see 3.5.2.2.2, in the case that the device is currently identifying itself.

3.5.2.3.1.1 Payload Format

The identify query response command payload SHALL be formatted as illustrated in Figure 3-9.

Figure 3-9. Format of Identify Query Response Command Payload

Octets	2
Data Type	uint16
Field Name	Timeout

3.5.2.3.1.2 Timeout Field

The Timeout field contains the current value of the *IdentifyTime* attribute, and specifies the length of time, in seconds, that the device will continue to identify itself.

3.5.2.3.1.3 Effect on Receipt

On receipt of this command, the device is informed of a device in the network which is currently identifying itself. This information MAY be particularly beneficial in situations where there is no commissioning tool. Note that there MAY be multiple responses.

3.5.3 Client

The client has no cluster specific attributes. The client generates the cluster specific commands detailed in 3.5.2.2, as required by the application. The client receives the cluster specific response commands detailed in 3.5.2.3.

3.6 Groups

3.6.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The stack specification provides the capability for group addressing. That is, any endpoint on any device MAY be assigned to one or more groups, each labeled with a 16-bit identifier (0x0001 to 0xfff7), which acts for all intents and purposes like a network address. Once a group is established, frames, sent using the APSDE-DATA.request primitive and having a DstAddrMode of 0x01, denoting group addressing, will be delivered to every endpoint assigned to the group address named in the DstAddr parameter of the outgoing APSDE-DATA.request primitive on every device in the network for which there are such endpoints.

Management of group membership on each device and endpoint is implemented by the APS, but the over-the-air messages that allow for remote management and commissioning of groups are defined here in the cluster library on the theory that, while the basic group addressing facilities are integral to the operation of the stack, not every device will need or want to implement this management cluster. Furthermore, the placement of the management commands here allows developers of proprietary profiles to avoid implementing the library cluster but still exploit group addressing.

Commands are defined here for discovering the group membership of a device, adding a group, removing a group and removing all groups.

Finally, the group cluster allows application entities to store a name string for each group to which they are assigned and to report that name string in response to a client request.

Note that configuration of group addresses for outgoing commands is achieved using the APS binding mechanisms, and is not part of this cluster.

As Groupcasts are made on a broadcast to all devices for which macRxOnWhenIdle = TRUE, sleeping end devices will not be able to benefit from the features of the Groups and Scenes server Cluster. For example, a door lock which would typically be a sleeping end device would not be able to receive the datagrams required to commission a scene or change for example, to a night scene. It is therefore not Mandatory but only optional to support the Groups and Scenes Server cluster if the device is a Sleeping end device (even when listed as Mandatory).

3.6.1.1 Revision History

Rev	Description
1	global mandatory *Cluster Revision* attribute added; CCB 1745 2100
2	CCB 2289

3.6.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	G

3.6.1.3 Cluster Identifiers

Identifier	Name
0x0004	Groups

[12]

[12] CCB 2289 removed Security clause about Permissions Configuration Table

3.6.2 Server

Each device that implements this cluster MAY be thought of as a group management server in the sense that it responds to information requests and configuration commands regarding the contents of its group table.

Note that, since these commands are simply data frames sent using the APSDE_SAP, they must be addressed with respect to device and endpoint. In particular, the destination device and endpoint of a group management command must be unambiguous at the time of the issuance of the primitive either because:

1. They are explicitly spelled out in the DstAddr and DstEndpoint parameters of the primitive.
2. They are not explicitly spelled out but MAY be derived from the binding table in the APS of the sending device.
3. Broadcast addressing is being employed, either with respect to the device address or the endpoint identifier.
4. Group addressing is being employed.

On receipt of a group cluster command, the APS will, at least conceptually, deliver the frame to each destination endpoint spelled out in the addressing portion of the APS header and, again conceptually speaking, the application entity resident at that endpoint will process the command and respond as necessary. From an implementation standpoint, of course, this MAY be done in a more economical way that does not involve duplication and separate processing, e.g., by providing a hook in the APS whereby group cluster commands could be delivered to a special application entity without duplication.

3.6.2.1 Dependencies

For correct operation of the 'Add group if identifying' command, any endpoint that implements the Groups server cluster SHALL also implement the Identify server cluster.

3.6.2.2 Attributes

The server supports the attribute shown in Table 3-36.

Table 3-36. Attributes of the Groups Server Cluster

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*NameSupport*	map8	x0000000	R	-	M

3.6.2.2.1 NameSupport Attribute

The most significant bit of the *NameSupport* attribute indicates whether or not group names are supported. A value of 1 indicates that they are supported, and a value of 0 indicates that they are not supported.

3.6.2.2.2 Group Names

Group names are between 0 and 16 characters long. Support of group names is optional, and is indicated by the *NameSupport* attribute. Group names, if supported, must be stored in a separate data structure managed by the application in which the entries correspond to group table entries.

3.6.2.3 Commands Received

The groups cluster is concerned with management of the group table on a device. In practice, the group table is managed by the APS and the table itself is available to the next higher layer as an AIB attribute. A command set is defined here and the implementation details of that command set in terms of the facilities provided by the APS is left up to the implementer of the cluster library itself.

The server side of the groups cluster is capable of receiving the commands listed in Table 3-37.

Table 3-37. Received Command IDs for the Groups Cluster

Command Identifier Field Value	Description	M/O
0x00	Add group	M
0x01	View group	M
0x02	Get group membership	M
0x03	Remove group	M
0x04	Remove all groups	M
0x05	Add group if identifying	M

3.6.2.3.1 Generic Usage Notes

On receipt of the *Add Group*, *View Group*, or *Remove Group* command frames via the groupcast or broadcast transmission service, no response SHALL be given.

3.6.2.3.2 Add Group Command

The Add Group command allows the sending device to add group membership in a particular group for one or more endpoints on the receiving device.

3.6.2.3.2.1 Payload Format

The Add Group command payload SHALL be formatted as illustrated in Figure 3-10.

Figure 3-10. Format of the Add Group Command Payload

Octets	2	Variable
Data Type	uint16	string
Field Name	Group ID	Group Name

3.6.2.3.2.2 Effect on Receipt

On receipt of this command, the device SHALL (if possible) add the Group ID and Group Name to its Group Table. If Group Name is not supported, the Group Name field SHALL be ignored. Except for the restrictions listed in 3.6.2.3.1, the device SHALL then generate an appropriate Add Group Response command indicating success or failure. See 3.6.2.4.1.

3.6.2.3.3 View Group Command

The view group command allows the sending device to request that the receiving entity or entities respond with a view group response command containing the application name string for a particular group.

3.6.2.3.3.1 Payload Format

The View Group command payload SHALL be formatted as illustrated in Figure 3-11:

Figure 3-11. Format of the View Group Command Payload

Octets	2
Data Type	uint16
Field Name	Group ID

3.6.2.3.3.2 Effect on Receipt

Except for the restrictions listed in 3.6.2.3.1, the device, on receipt of this command, SHALL generate an appropriate View Group Response command 3.6.2.4.2.

3.6.2.3.4 Get Group Membership Command

The get group membership command allows the sending device to inquire about the group membership of the receiving device and endpoint in a number of ways.

3.6.2.3.4.1 Payload Format

The get group membership command payload SHALL be formatted as illustrated in Figure 3-12.

Figure 3-12. Format of Get Group Membership Command Payload

Octets	1	Variable
Data Type	uint8	List of 16-bit integers
Field Name	Group count	Group list

3.6.2.3.4.2 Effect on Receipt

On receipt of the get group membership command, each receiving entity SHALL respond with group membership information using the get group membership response frame as follows:

If the group count field of the command frame has a value of 0 indicating that the group list field is empty, the entity SHALL respond with all group identifiers of which the entity is a member.

If the group list field of the command frame contains at least one group of which the entity is a member, the entity SHALL respond with each entity group identifier that match a group in the group list field.

If the group count is non-zero, and the group list field of the command frame does not contain any group of which the entity is a member, the entity SHALL only respond if the command is unicast. The response SHALL return a group count of zero.

3.6.2.3.5 Remove Group Command

The remove group command allows the sender to request that the receiving entity or entities remove their membership, if any, in a particular group.

Note that if a group is removed the scenes associated with that group SHOULD be removed.

3.6.2.3.5.1 Payload Format

The Remove Group command payload SHALL be formatted as illustrated in Figure 3-13.

Figure 3-13. Format of the Remove Group Command Payload

Octets	2
Data Type	uint16
Field Name	Group ID

3.6.2.3.5.2 Effect on Receipt

On receipt of this command, the device SHALL (if possible) remove the Group ID and Group Name from its Group Table. Except for the restrictions listed in 3.6.2.3.1, the device SHALL then generate an appropriate Remove Group Response command indicating success or failure. See 3.6.2.4.4.

3.6.2.3.6 Remove All Groups Command

The remove all groups command allows the sending device to direct the receiving entity or entities to remove all group associations.

Note that removing all groups necessitates the removal of all associated scenes as well. (Note: scenes not associated with a group need not be removed).

3.6.2.3.6.1 Payload Format

The Remove All Groups command has no payload.

3.6.2.3.6.2 Effect on Receipt

On receipt of this command, the device SHALL remove all groups on this endpoint from its Group Table.

3.6.2.3.7 Add Group If Identifying Command

The add group if identifying command allows the sending device to add group membership in a particular group for one or more endpoints on the receiving device, on condition that it is identifying itself. Identifying functionality is controlled using the identify cluster, (see 3.5).

This command might be used to assist configuring group membership in the absence of a commissioning tool.

3.6.2.3.7.1 Payload Format

The Add Group If Identifying command payload SHALL be formatted as illustrated in Figure 3-14.

Figure 3-14. Add Group If Identifying Command Payload

Octets	2	Variable
Data Type	uint16	string
Field Name	Group ID	Group Name

3.6.2.3.7.2 Effect on Receipt

On receipt of this command, the device SHALL first check whether it is currently identifying itself. If so then the device SHALL (if possible) add the Group ID and Group Name to its Group Table. If the device it not currently identifying itself then no action SHALL be taken.

No response is defined as this command is EXPECTED to be multicast or broadcast.

3.6.2.4 Commands Generated

The commands generated by the server side of the group cluster, as listed in Table 3-38, are responses to the received commands listed in sub-clause 3.6.2.3.

Table 3-38. Generated Command IDs for the Groups Cluster

Command Identifier Field Value	Description	M/O
0x00	Add group response	M
0x01	View group response	M
0x02	Get group membership response	M
0x03	Remove group response	M

Note: There is no need for a response to the Remove all Groups command, as, at an application level, this command always succeeds.

3.6.2.4.1 Add Group Response Command

The add group response is sent by the groups cluster server in response to an add group command.

3.6.2.4.1.1 Payload Format

The Add Group Response command payload SHALL be formatted as illustrated in Figure 3-15.

Figure 3-15. Format of the Add Group Response Command Payload

Octets	1	2
Data Type	enum8	uint16
Field Name	Status	Group ID

3.6.2.4.1.2 When Generated

This command is generated in response to a received Add Group command. The Status field is set to SUCCESS, DUPLICATE_EXISTS, or INSUFFICIENT_SPACE as appropriate. The Group ID field is set to the Group ID field of the received Add Group command.

3.6.2.4.2 View Group Response Command

The view group response command is sent by the groups cluster server in response to a view group command.

3.6.2.4.2.1 Payload Format

The View Group Response command payload SHALL be formatted as illustrated in Figure 3-16.

Figure 3-16. Format of the View Group Response Command Payload

Octets	1	2	Variable
Data Type	enum8	uint16	string
Field Name	Status	Group ID	Group Name

3.6.2.4.2.2 When Generated

This command is generated in response to a received View Group command. The Status field is set to SUCCESS or NOT_FOUND as appropriate. The Group ID field is set to the Group ID field of the received View Group command. If the status is SUCCESS, and group names are supported, the Group Name field is set to the Group Name associated with that Group ID in the Group Table; otherwise it is set to the null (empty) string, i.e., a single octet of value 0.

3.6.2.4.3 Get Group Membership Response Command

The get group membership response command is sent by the groups cluster server in response to a get group membership command.

3.6.2.4.3.1 Payload Format

The payload of the get group membership response command is formatted as shown in Figure 3-17.

Figure 3-17. Format of the Get Group Membership Response Command Payload

Octets	1	1	Variable
Data Type	uint8	uint8	List of 16-bit group ID
Field Name	Capacity	Group count	Group list

The fields of the get group membership response command have the following semantics:

The Capacity field SHALL contain the remaining capacity of the group table of the device. The following values apply:

0	No further groups MAY be added.
0 < Capacity < 0xfe	Capacity holds the number of groups that MAY be added
0xfe	At least 1 further group MAY be added (exact number is unknown)
0xff	It is unknown if any further groups MAY be added

The Group count field SHALL contain the number of groups contained in the group list field.

The Group list field SHALL contain the identifiers either of all the groups in the group table (in the case where the group list field of the received get group membership command was empty) or all the groups from the group list field of the received get group membership command which are in the group table. If the total number of groups will cause the maximum payload length of a frame to be exceeded, then the Group list field shall contain only as many groups as will fit.

3.6.2.4.3.2 When Generated

See Get Group Membership Command 3.6.2.3.4.2 Effect on Receipt.

3.6.2.4.4 Remove Group Response Command

The remove group response command is generated by an application entity in response to the receipt of a remove group command.

3.6.2.4.4.1 Payload Format

The Remove Group Response command payload SHALL be formatted as illustrated in Figure 3-18.

Figure 3-18. Format of Remove Group Response Command Payload

Octets	1	2
Data Type	enum8	uint16
Field Name	Status	Group ID

3.6.2.4.4.2 When Generated

This command is generated in response to a received Remove Group command. The Status field is set to SUCCESS or NOT_FOUND as appropriate. The Group ID field is set to the Group ID field of the received Remove Group command.

3.6.3 Client

The Client cluster has no cluster specific attributes. The client generates the cluster specific commands detailed in 3.6.2.3. The client receives the cluster specific response commands detailed in 3.6.2.4.

3.7 Scenes

3.7.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The scenes cluster provides attributes and commands for setting up and recalling scenes. Each scene corresponds to a set of stored values of specified attributes for one or more clusters on the same end point as the scenes cluster.

In most cases scenes are associated with a particular group ID. Scenes MAY also exist without a group, in which case the value 0x0000 replaces the group ID. Note that extra care is required in these cases to avoid a scene ID collision, and that commands related to scenes without a group MAY only be unicast, i.e., they MAY not be multicast or broadcast.

3.7.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added; CCB 1745
2	Recall Scene Transition Time field

3.7.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	S	Type 1 (client to server)

3.7.1.3 Cluster Identifiers

Identifier	Name
0x0005	Scenes

3.7.2 Server

3.7.2.1 Dependencies

Any endpoint that implements the Scenes server cluster SHALL also implement the Groups server cluster.

3.7.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 3-39.

Table 3-39. Scenes Attribute Sets

Attribute Set Identifier	Description
0x000	Scene Management Information

3.7.2.2.1 Scene Management Information Attribute Set

The Scene Management Information attribute set contains the attributes summarized in Table 3-40.

Table 3-40. Scene Management Information Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0000	*SceneCount*	uint8	0x00 to 0xff (see 3.7.2.3.2)	R	0x00	M
0x0001	*CurrentScene*	uint8	0x00 to 0xff (see 3.7.2.3.2)	R	0x00	M
0x0002	*CurrentGroup*	uint16	0x0000 to 0xfff7	R	0x00	M
0x0003	*SceneValid*	bool	0x00 to 0x01	R	0x00	M
0x0004	*NameSupport*	map8	x0000000	R	-	M
0x0005	*LastConfiguredBy*	EUI64	-	R	-	O

3.7.2.2.1.1 SceneCount Attribute

The *SceneCount* attribute specifies the number of scenes currently in the device's scene table.

3.7.2.2.1.2 *CurrentScene* Attribute

The *CurrentScene* attribute holds the Scene ID of the scene last invoked.

3.7.2.2.1.3 *CurrentGroup* Attribute

The *CurrentGroup* attribute holds the Group ID of the scene last invoked, or 0x0000 if the scene last invoked is not associated with a group.

3.7.2.2.1.4 SceneValid Attribute

The *SceneValid* attribute indicates whether the state of the device corresponds to that associated with the *CurrentScene* and *CurrentGroup* attributes. TRUE indicates that these attributes are valid, FALSE indicates that they are not valid.

Before a scene has been stored or recalled, this attribute is set to FALSE. After a successful Store Scene or Recall Scene command it is set to TRUE. If, after a scene is stored or recalled, the state of the device is modified, this attribute is set to FALSE.

3.7.2.2.1.5 NameSupport Attribute

The most significant bit of the *NameSupport* attribute indicates whether or not scene names are supported. A value of 1 indicates that they are supported, and a value of 0 indicates that they are not supported.

3.7.2.2.1.6 *LastConfiguredBy* Attribute

The *LastConfiguredBy* attribute is 64 bits in length and specifies the IEEE address of the device that last configured the scene table.

The value 0xffffffffffffffff indicates that the device has not been configured, or that the address of the device that last configured the scenes cluster is not known.

3.7.2.3 Scene Table

The scene table is used to store information for each scene capable of being invoked on a device. Each scene is defined for a particular group.

The fields of each scene table entry consist of a number of sets. The base set consists of the first four fields of Table 3-41. A set of extension fields can be added by each additional cluster implemented on a device.

Table 3-41. Fields of a Scene Table Entry

Field	Type	Valid Range	Description
Scene group ID	uint16	0x0000 to 0xfff7	The group ID for which this scene applies, or 0x0000 if the scene is not associated with a group.
Scene ID	uint8	0x00 to 0xff (see 3.7.2.3.2)	The identifier, unique within this group, which is used to identify this scene.
Scene name	string	0 to 16 characters	The name of the scene (optional)
Scene transition time	uint16	0x0000 to 0xffff	The amount of time, in seconds, it will take for the device to change from its current state to the requested scene.
Extension field sets	Variable	Variable	See the Scene Table Extensions subsections of individual clusters. Each extension field set holds a set of values of attributes for a cluster implemented on the device. The sum of all such sets defines a scene.
TransitionTime100ms	uint8	0x00 to 0x09	Together with the scene transition time element, this allows the transition time to be specified in tenths of a second.

3.7.2.3.1 Scene Names

Scene names are between 0 and 16 characters long. Support of scene names is optional, and is indicated by the *NameSupport* attribute. If scene names are not supported, any commands that write a scene name SHALL simply discard the name, and any command that returns a scene names SHALL return the null string.

3.7.2.3.2 Maximum Number of Scenes

The number of scenes capable of being stored in the table is defined by the profile in which this cluster is used. The default maximum, in the absence of specification by the profile, is 16.

3.7.2.4 Commands Received

The received command IDs for the Scenes cluster are listed in Table 3-42.

Table 3-42. Received Command IDs for the Scenes Cluster

Command Identifier Field Value	Description	M/O
0x00	Add Scene	M
0x01	View Scene	M
0x02	Remove Scene	M
0x03	Remove All Scenes	M
0x04	Store Scene	M
0x05	Recall Scene	M
0x06	Get Scene Membership	M
0x40	Enhanced Add Scene	O
0x41	Enhanced View Scene	O
0x42	Copy Scene	O

3.7.2.4.1 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*, *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.

3.7.2.4.2 Add Scene Command

3.7.2.4.2.1 Payload Format

The payload SHALL be formatted as illustrated in Figure 3-19.

Figure 3-19. Format of the Add Scene Command Payload

Octets	2	1	2	Variable	Variable
Data Type	uint16	uint8	uint16	string	Variable (multiple types)
Field Name	Group ID	Scene ID	Transition time	Scene Name	Extension field sets, one per cluster

The format of each extension field set is a 16 bit field carrying the cluster ID, followed by an 8 bit length field and the set of scene extension fields specified in the relevant cluster. The length field holds the length in octets of that extension field set.

Extension field sets =
{{clusterId 1, length 1, {extension field set 1}}, {clusterId 2, length 2, {extension field set 2}} ...}.

The attributes included in the extension field set for each cluster are defined in the specification for that cluster in this document (the Cluster Library). The field set consists of values for these attributes concatenated together, in the order given in the cluster specification, with no attribute identifiers or data type indicators.

For forward compatibility, reception of this command SHALL allow for the possible future addition of other attributes to the trailing ends of the lists given in the cluster specifications (by ignoring them). Similarly, it SHALL allow for one or more attributes to be omitted from the trailing ends of these lists (see 3.7.2.4.7.2).

It is not mandatory for a field set to be included in the command for every cluster on that endpoint that has a defined field set. Extension field sets MAY be omitted, including the case of no field sets at all.

3.7.2.4.2.2 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 ID and Group ID, it SHALL overwrite it, (i.e., it SHALL first remove all information included in the original scene entry).

Except for the restrictions in 3.7.2.4.1, the device SHALL then generate an appropriate Add Scene Response command indicating success or failure. See 3.7.2.5.1.

3.7.2.4.3 View Scene Command

3.7.2.4.3.1 Payload Format

The payload SHALL be formatted as illustrated in Figure 3-20.

Figure 3-20. Format of the View Scene Command Payload

Octets	2	1
Data Type	uint16	uint8
Field Name	Group ID	Scene ID

3.7.2.4.3.2 Effect on Receipt

On receipt of this command, except for the restrictions in 3.7.2.4.1, the device SHALL generate an appropriate View Scene Response command. See 3.7.2.5.2.

3.7.2.4.4 Remove Scene Command

3.7.2.4.4.1 Payload Format

The Remove Scene command payload SHALL be formatted as illustrated in Figure 3-21.

Figure 3-21. Format of the Remove Scene Command Payload

Octets	2	1
Data Type	uint16	uint8
Field Name	Group ID	Scene ID

3.7.2.4.4.2 Effect on Receipt

On receipt of this command, the device SHALL (if possible) remove from its Scene Table the entry with this Scene ID and group ID. If the command was addressed to a single device (not a group) then it SHALL generate an appropriate Remove Scene Response command indicating success or failure. See 3.7.2.5.3.

3.7.2.4.5 Remove All Scenes Command

3.7.2.4.5.1 Payload Format

The Remove All Scenes command payload SHALL be formatted as illustrated in Figure 3-22.

Figure 3-22. Format of the Remove All Scenes Command Payload

Octets	2
Data Type	uint16
Field Name	Group ID

3.7.2.4.5.2 Effect on Receipt

On receipt of this command, the device SHALL, if possible, remove from its Scene Table all entries with this Group ID. If the command was addressed to a single device (not to a group) it SHALL then generate an appropriate Remove All Scenes Response command indicating success or failure. See 3.7.2.5.3.3.

3.7.2.4.6 Store Scene Command

3.7.2.4.6.1 Payload Format

The Store Scene command payload SHALL be formatted as illustrated in Figure 3-23.

Figure 3-23. Format of the Store Scene Command Payload

Octets	2	1
Data Type	uint16	uint8
Field Name	Group ID	Scene ID

3.7.2.4.6.2 Effect on Receipt

On receipt of this command, the device SHALL (if possible) add an entry to the Scene Table with the Scene ID and Group ID given in the command, and all extension field sets corresponding to the current state of other clusters on the device.

If an entry already exists with the same Scene ID and Group ID, as a result of a previous Add Scene command, the extension field sets are overwritten (i.e., completely replaced) with the current values of the attributes that are extension fields on clusters that are on the same endpoint, but the transition time field and the scene name field are left unaltered. If no such entry exists, the transition time field SHALL be set to 0, and the scene name field SHALL be set to the null string.

Note that, accordingly, if a scene to be stored requires a transition time field and/ or a scene name field, these must be set up by a prior Add Scene command, e.g., with no scene extension field sets.

If the Group ID field is not zero, and the device is not a member of this group, the scene will not be added.

If the command was addressed to a single device (not to a group) then it SHALL generate an appropriate Store Scene Response command indicating success or failure. See 3.7.2.5.3.6.

3.7.2.4.7 Recall Scene Command

3.7.2.4.7.1 Payload Format

The Recall Scene command payload SHALL be formatted as illustrated in Figure 3-24.

Figure 3-24. Format of the Recall Scene Command Payload

Octets	2	1	0/2
Data Type	uint16	uint8	uint16
Field Name	Group ID	Scene ID	Transition Time[13]

3.7.2.4.7.2 Effect on Receipt

On receipt of this command, the device SHALL (if possible) locate the entry in its Scene Table with the Group ID and Scene ID given in the command. For each other cluster on the device, it SHALL then retrieve any corresponding extension fields from the Scene Table and set the attributes and corresponding state of the cluster accordingly.

If there is no extension field set for a cluster, the state of that cluster SHALL remain unchanged. If an extension field set omits the values of any trailing attributes, the values of these attributes SHALL remain unchanged.

[13] NFR Add Transition Time to Recall Scene

If the Transition Time field is present in the command payload and its value is not equal to 0xFFFF, this field SHALL indicate the transition time in 1/10ths of a second. In all other cases (command payload field not present or value equal to 0xFFFF), The scene transition time field of the Scene Table entry SHALL indicate the transition time. The transition time determines how long the transition takes from the old cluster state to the new cluster state. It is recommended that, where possible (e.g., it is not possible for attributes with Boolean data type), a gradual transition SHOULD take place from the old to the new state over this time. However, the exact transition is manufacturer dependent.

This command does not result in a response command.

3.7.2.4.8 Get Scene Membership Command

The Get Scene Membership command can be used to find an unused scene number within the group when no commissioning tool is in the network, or for a commissioning tool to get used scenes for a group on a single device or on all devices in the group.

3.7.2.4.8.1 Payload Format

The Get Scene Membership command payload SHALL be formatted as illustrated in Figure 3-25.

Figure 3-25. Format of Get Scene Membership Command Payload

Octets	2
Data Type	uint16
Field Name	Group ID

3.7.2.4.8.2 Effect on Receipt

On receipt of this command, the device SHALL if addressed to a single device generate an appropriate Get Scene Membership Response command; otherwise it SHALL only generate an appropriate Get Scene Membership Response command if an entry within the Scene Table corresponds to the Group ID. See 3.7.2.5.3.9.

3.7.2.4.9 Enhanced Add Scene Command

The *Enhanced Add Scene* command allows a scene to be added using a finer scene transition time than the *Add Scene* command.

The payload of this command SHALL be formatted in the same way as the *Add Scene* command, specified in the ZCL S*cenes* cluster, with the following difference:

The *Transition Time* field SHALL be measured in tenths of a second rather than in seconds.

3.7.2.4.9.1 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 S*cenes* cluster server SHALL then generate and transmit an *Enhanced Add Scene Response* command back to the originator. Otherwise, no response SHALL be given.

3.7.2.4.10 Enhanced View Scene Command

The *Enhanced View Scene* command allows a scene to be retrieved using a finer scene transition time than the *View Scene* command.

The payload of this command SHALL be formatted in the same way as the *View Scene* command.

3.7.2.4.10.1 Effect on Receipt

On receipt of this command, the device SHALL generate an appropriate *enhanced view scene response* command.

3.7.2.4.11 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 3-26.

Figure 3-26. Format of the Copy Scene Command

Octets	1	2	1	2	1
Data Type	uint8	uint16	uint8	uint16	uint8
Field Name	Mode	Group identifier from	Scene identifier from	Group identifier to	Scene identifier to

3.7.2.4.11.1 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 3-27.

Figure 3-27. Format of the Mode Field of the Copy Scene Command

Bits: 0	**Bits: 1-7**
Copy All Scenes	Reserved

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.

3.7.2.4.11.2 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.

3.7.2.4.11.3 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.

3.7.2.4.11.4 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.

3.7.2.4.11.5 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.

3.7.2.4.11.6 Effect on Receipt

On receipt of the *Copy Scene* command, if the *Copy All Scenes* sub-field of the M*ode* 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.

3.7.2.5 Commands Generated

The generated command IDs for the Scenes cluster are listed in Table 3-43.

Table 3-43. Generated Command IDs for the Scenes Cluster

Command Identifier Field Value	Description	M/O
0x00	Add Scene Response	M
0x01	View Scene Response	M
0x02	Remove Scene Response	M
0x03	Remove All Scenes Response	M
0x04	Store Scene Response	M
0x06	Get Scene Membership Response	M
0x40	Enhanced Add Scene Response	O
0x41	Enhanced View Scene Response	O
0x42	Copy Scene Response	O

3.7.2.5.1 Add Scene Response Command

3.7.2.5.1.1 Payload Format

The Add Scene Response command payload SHALL be formatted as illustrated in Figure 3-28.

Figure 3-28. Format of the Add Scene Response Command Payload

Octets	1	2	1
Data Type	enum8	uint16	uint8
Field Name	Status	Group ID	Scene ID

3.7.2.5.1.2 When Generated

This command is generated in response to a received Add Scene command 3.7.2.4.2. The Status field is set to SUCCESS, INSUFFICIENT_SPACE or INVALID_FIELD (the group is not present in the Group Table) as appropriate. The Group ID and Scene ID fields are set to the corresponding fields of the received Add Scene command.

3.7.2.5.2 View Scene Response Command

3.7.2.5.2.1 Payload Format

The View Scene Response command payload SHALL be formatted as illustrated in Figure 3-29.

Figure 3-29. Format of the View Scene Response Command Payload

Octets	1	2	1	0 / 2	0 / Variable	0 / Variable
Data Type	enum8	uint16	uint8	uint16	string	Variable (multiple types)
Field Name	Status	Group ID	Scene ID	Transition time	Scene Name	Extension field sets, one per cluster

The format of each extension field set is a 16 bit field carrying the cluster ID, followed by an 8 bit data length field and the set of scene extension fields specified in the relevant cluster. These fields are concatenated together in the order given in the cluster.

Extension field sets =

{{clusterId 1, length 1, {extension field set 1}}, {clusterId 2, length 2, {extension field set 2}},}.

3.7.2.5.2.2 When Generated

This command is generated in response to a received View Scene command 3.7.2.4.3.

The entry in the Scene Table with Scene ID and Group ID given in the received 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 ID and Scene ID fields are set to the corresponding fields in the received 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.

3.7.2.5.3 Remove Scene Response Command

3.7.2.5.3.1 Payload Format

The Remove Scene Response command payload SHALL be formatted as illustrated in Figure 3-30.

Figure 3-30. Format of Remove Scene Response Command Payload

Octets	1	2	1
Data Type	enum8	uint16	uint8
Field Name	Status	Group ID	Scene ID

3.7.2.5.3.2 When Generated

This command is generated in response to a received Remove Scene command3.7.2.4.4. 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 ID and Scene ID fields are set to the corresponding fields of the received Remove Scene command.

3.7.2.5.3.3 Remove All Scenes Response Command

3.7.2.5.3.4 Payload Format

The Remove All Scenes Response command payload SHALL be formatted as illustrated in Figure 3-31.

Figure 3-31. Format of the Remove All Scenes Response Command Payload

Octets	1	2
Data Type	enum8	uint16
Field Name	Status	Group ID

3.7.2.5.3.5 When Generated

This command is generated in response to a received Remove All Scenes command, see 3.7.2.4.5. The Status field is set to SUCCESS or INVALID_FIELD (the group is not present in the Group Table) as appropriate. The Group ID field is set to the corresponding field of the received Remove All Scenes command.

3.7.2.5.3.6 Store Scene Response Command

3.7.2.5.3.7 Payload Format

The Store Scene Response command payload SHALL be formatted as illustrated in Figure 3-32.

Figure 3-32. Format of the Store Scene Response Command Payload

Octets	1	2	1
Data Type	enum8	uint16	uint8
Field Name	Status	Group ID	Scene ID

3.7.2.5.3.8 When Generated

This command is generated in response to a received Store Scene command 3.7.2.4.6. The Status field is set to SUCCESS, INSUFFICIENT_SPACE or INVALID_FIELD (the group is not present in the Group Table) as appropriate. The Group ID and Scene ID fields are set to the corresponding fields of the received Store Scene command.

3.7.2.5.3.9 Get Scene Membership Response Command

3.7.2.5.3.10 Payload Format

The Get Scene Membership Response command payload SHALL be formatted as illustrated in Figure 3-33.

Figure 3-33. Format of the Get Scene Membership Response CommandPayload

Octets	1	1	2	0/1	Variable
Data Type	enum8	uint8	uint16	uint8	uint8 x N
Field Name	Status	Capacity	Group ID	Scene count	Scene list

The fields of the get scene membership response command have the following semantics:

The Capacity field SHALL contain the remaining capacity of the scene table of the device (for all groups). The following values apply:

0	No further scenes MAY be added.
0 < Capacity < 0xfe	Capacity holds the number of scenes that MAY be added
0xfe	At least 1 further scene MAY be added (exact number is unknown)
0xff	It is unknown if any further scenes MAY be added

The Status field SHALL contain SUCCESS or INVALID_FIELD (the group is not present in the Group Table) as appropriate.

The Group ID field SHALL be set to the corresponding field of the received Get Scene Membership command.

If the status is not SUCCESS, then the Scene count and Scene list field are omitted, else

The Scene count field SHALL contain the number of scenes contained in the Scene list field.

The Scene list field SHALL contain the identifiers of all the scenes in the scene table with the corresponding Group ID. If the total number of scenes associated with this Group ID will cause the maximum payload length of a frame to be exceeded, then the Scene list field shall contain only as many scenes as will fit.

3.7.2.5.3.11 When Generated

This command is generated in response to a received Get Scene Membership command, 3.7.2.4.8.

3.7.2.5.4 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.

3.7.2.5.5 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, with the following difference:

The *Transition Time* field SHALL be measured in tenths of a second rather than in seconds.

3.7.2.5.5.1 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.

3.7.2.5.6 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 3-34.

Figure 3-34. Format of the Copy Scene Response Command

Octets	1	2	1
Data Type	uint8	uint16	uint8
Field Name	Status	Group identifier from	Scene identifier from

3.7.2.5.6.1 3.7.2.5.9.1 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 3-44.

Table 3-44. Values of the Status Field of the Copy Scene Response Command

Status Field Value[14]	**Description**
SUCCESS	Success
INVALID_FIELD	Invalid scene specified
INSUFFICIENT_SPACE	Insufficient space in the scene table

3.7.2.5.6.2 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.

3.7.2.5.6.3 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.

[14] See Chapter 2 for an enumerated list of status values.

3.7.2.5.6.4 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.

3.7.3 Client

The Client cluster has no cluster specific attributes. The client generates the cluster specific commands detailed in 3.7.2.4, as required by the application. The client receives the cluster specific response commands detailed in 3.7.2.5.

3.8 On/Off

3.8.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

Attributes and commands for switching devices between ‘On’ and ‘Off’ states.

3.8.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added; CCB 1555
2	ZLO 1.0: StartUpOnOff

3.8.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	OO	Type 1 (client to server)

3.8.1.3 Cluster Identifiers

Identifier	PICS Code	Name
0x0006	OO	On/Off

3.8.2 Server

3.8.2.1 Dependencies

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.

3.8.2.2 Attributes

The server supports the attributes shown in Table 3-45.

Table 3-45. Attributes of the On/Off Server Cluster

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*OnOff*	bool	0x00 to 0x01	RPS	0x00	M
0x4000	*GlobalSceneControl*	bool	0x00 to 0x01	R	0x01	O
0x4001	*OnTime*	uint16	0x0000 to 0xffff	RW	0x0000	O
0x4002	*OffWaitTime*	uint16	0x0000 to 0xffff	RW	0x0000	O
0x4003	*StartUpOnOff*[15]	enum8	0x00 to 0xff	RW	*MS*	O

3.8.2.2.1 *OnOff* Attribute

The *OnOff* attribute has the following values: 0 = Off, 1 = On.

3.8.2.2.2 *GlobalSceneControl* Attribute

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

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

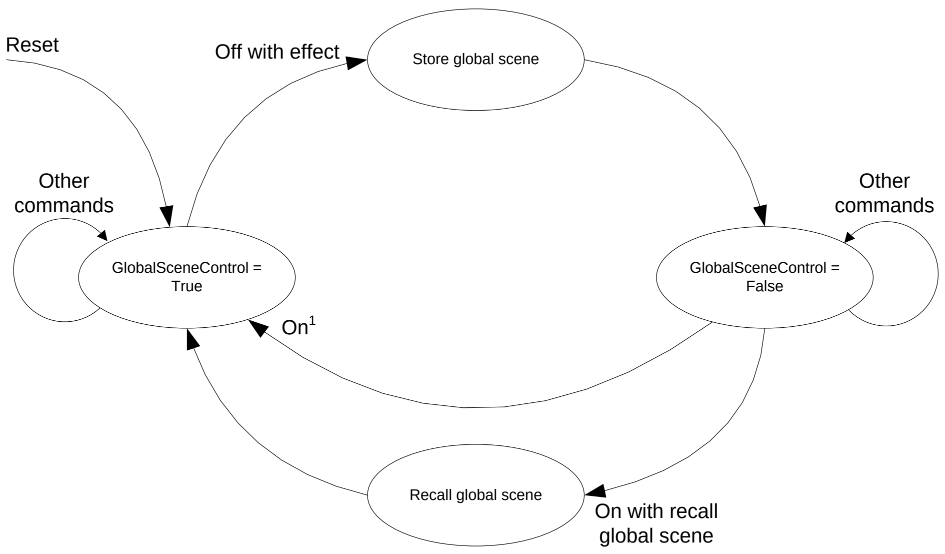
The *GlobalSceneControl* attribute SHALL be set to TRUE after the reception of a command which causes the *OnOff* attribute to be set to TRUE, such as a standard *On* command, a *Move to level (with on/off)* command, a *Recall scene* command or a *On with recall global scene* command (see Section 3.8.2.3.5).

The *GlobalSceneControl* attribute is set to FALSE after reception of a *Off with effect* command.

[15] ZLO 1.0

These concepts are illustrated in Figure 3-35.

Figure 3-35. State Behavior of Store and Recall Global Scene



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

3.8.2.2.3 *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 device SHALL remain in its current state.

3.8.2.2.4 *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 device 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 device SHALL remain in its current state.

3.8.2.2.5 *StartUpOnOff* Attribute

The *StartUpOnOff* attribute SHALL define the desired startup behavior of a lamp device when it is supplied with power and this state SHALL be reflected in the *OnOff* attribute. The values of the *StartUpOnOff* attribute are listed below.

Table 3-46. Values of the *StartUpOnOff* Attribute

Value	Action on power up
0x00	Set the *OnOff* attribute to 0 (off).
0x01	Set the *OnOff* attribute to 1 (on).
0x02	If the previous value of the *OnOff* attribute is equal to 0, set the *OnOff* attribute to 1. If the previous value of the *OnOff* attribute is equal to 1, set the *OnOff* attribute to 0 (toggle).
0x03 to 0xfe	These values are reserved. No action.
0xff	Set the *OnOff* attribute to its previous value.

3.8.2.3 Commands Received

The command IDs for the *On/Off* cluster are listed below.

Table 3-47. Command IDs for the On/Off Cluster

ID	Description	M/O
0x00	Off	M
0x01	On	M
0x02	Toggle	M
0x40	Off with effect	O
0x41	On with recall global scene	O
0x42	On with timed off	O

3.8.2.3.1 Off Command

This command does not have a payload.

3.8.2.3.1.1 Effect on Receipt

On receipt of this command, a device SHALL enter its 'Off' state. This state is device dependent, but it is recommended that it is used for power off or similar functions. On receipt of the *Off* command, the *OnTime* attribute SHALL be set to 0x0000.

3.8.2.3.2 On Command

This command does not have a payload.

3.8.2.3.2.1 Effect on Receipt

On receipt of this command, a device SHALL enter its ‘On’ state. This state is device dependent, but it is recommended that it is used for power on or similar functions. 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.

3.8.2.3.3 Toggle Command

This command does not have a payload.

3.8.2.3.3.1 Effect on Receipt

On receipt of this command, if a device is in its ‘Off’ state it SHALL enter its ‘On’ state. Otherwise, if it is in its ‘On’ state it SHALL enter its ‘Off’ state. 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.

3.8.2.3.4 Off With Effect Command

The *Off With Effect* command allows devices to be turned off using enhanced ways of fading.

The payload of this command SHALL be formatted as illustrated in Figure 3-36.

Figure 3-36. Format of the Off With Effect Command

Octets	1	1
Data Type	uint8	uint8
Field Name	Effect identifier	Effect variant

3.8.2.3.4.1 Effect Identifier Field

The *Effect Identifier* field is 8-bits in length and specifies the fading effect to use when switching the device off. This field SHALL contain one of the non-reserved values listed in Table 3-48.

Table 3-48. 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 to 0xff	Reserved

3.8.2.3.4.2 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 nonreserved values listed in Table 3-49.

Table 3-49. 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

Effect Identifier Field Value	Effect Variant Field Value	Description
	0x01	No fade
	0x02	50% dim down in 0.8 seconds then fade to off in 12 seconds
	0x03 to 0xff	Reserved
0x01	0x00 (default)	20% dim up in 0.5s then fade to off in 1 second
	0x01 to 0xff	Reserved
0x02 to 0xff	0x00 to 0xff	Reserved

3.8.2.3.4.3 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.

3.8.2.3.5 On With Recall Global Scene Command

The *On With Recall Global Scene* command allows the recall of the settings when the device was turned off.

The *On With Recall Global Scene* command SHALL have no parameters.

3.8.2.3.5.1 Effect on Receipt

On receipt of the *On With Recall Global Scene* command, if the *GlobalSceneControl* attribute is equal to TRUE, the application on the associated endpoint SHALL discard the 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 Addition, if the value of the *OnTime* attribute is equal to 0x0000, the device SHALL then set the *OffWaitTime* attribute to 0x0000.

3.8.2.3.6 On With Timed Off Command

The *On With Timed Off* command allows devices to be turned on for a specific duration with a guarded off duration so that SHOULD the device be subsequently switched off, further *On With Timed Off* commands, received during this time, are prevented from turning the devices back on. Note that the device 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 3-37.

Figure 3-37. Format of the On With Timed Off Command

Octets	1	2	2
Data Type	uint8	uint16	uint16
Field Name	On/off Control	On Time	Off Wait Time

3.8.2.3.6.1 On/Off Control Field

The *On/Off Control* field is 8-bits in length and contains information on how the device is to be operated. This field SHALL be formatted as illustrated in Figure 3-38.

Figure 3-38. Format of the On/Off Control Field of the On With Timed Off Command

Bits: 0	1-7
Accept Only When On	Reserved

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.

3.8.2.3.6.2 On Time Field

The *On Time* field is 16 bits in length and specifies the length of time (in 1/10ths second) that the device 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 to 0xfffe.

3.8.2.3.6.3 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 device SHALL remain "off", i.e., with its *OnOff* attribute equal to 0x00, and guarded to prevent an on command turning the device back "on". This field SHALL be specified in the range 0x0000 to 0xfffe.

3.8.2.3.6.4 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).

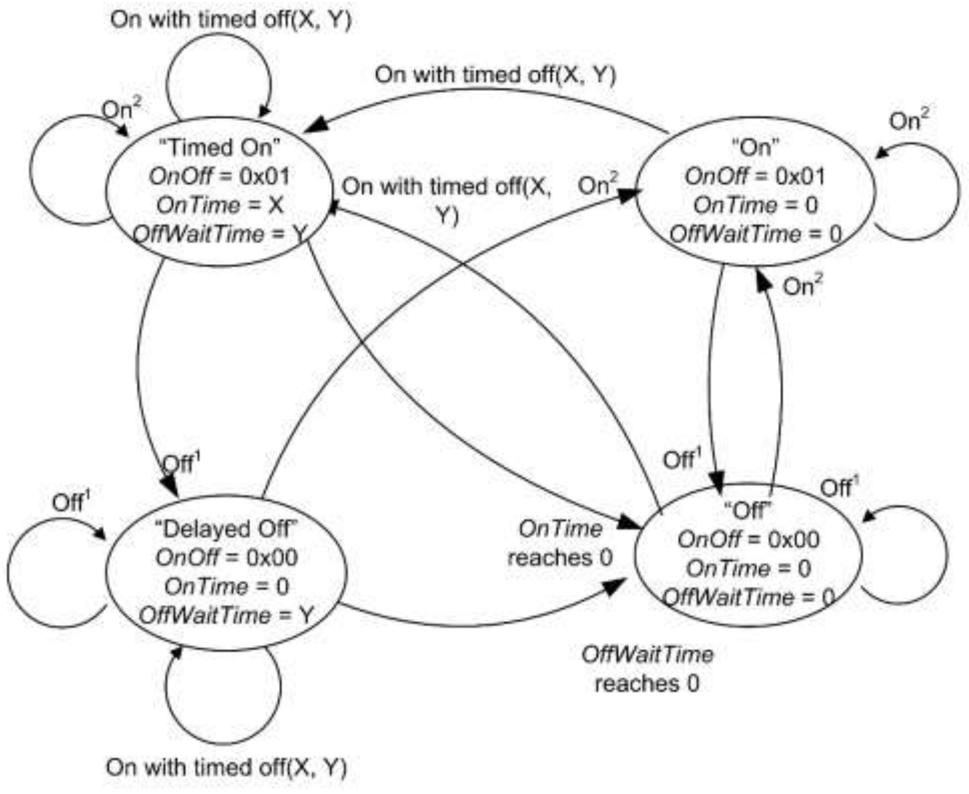
If the values of the *OnTime* and *OffWaitTime* attributes are both less than 0xffff, the device SHALL then update the device every 1/10th second until both the *OnTime* and *OffWaitTime* attributes are equal to 0x0000, as follows:

- 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.

3.8.2.4 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 3-39. 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.

Figure 3-39. On/Off Cluster Operation State Machine



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.

3.8.2.5 Commands Generated

The server generates no commands.

3.8.2.6 Scene Table Extensions

If the Scenes server cluster (11) is implemented, the following extension field is added to the Scenes table:

OnOff

3.8.2.7 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes command and according to the minimum and maximum reporting interval settings described in Chapter 2, Foundation. The following attribute SHALL be reported:

OnOff

3.8.3 Client

The client has no cluster specific attributes. The client generates the cluster specific commands received by the server (see 3.8.2.3) , as required by the application. No cluster specific commands are received by the client.

3.9 On/Off Switch Configuration

3.9.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

Attributes and commands for configuring On/Off switching devices.

3.9.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added

3.9.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	OOSC	Type 2 (server to client)

3.9.1.3 Cluster Identifiers

Identifier	Name
0x0007	On /Off Switch Configuration

3.9.2 Server

3.9.2.1 Dependencies

Any endpoint that implements this server cluster SHALL also implement the On/Off client cluster.

3.9.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 3-50.

Table 3-50. On/Off Switch Configuration Attribute Sets

Attribute Set Identifier	Description
0x000	Switch Information
0x001	Switch Settings

3.9.2.2.1 Switch Information Attribute Set

The switch information attribute set contains the attributes summarized in Table 3-51.

Table 3-51. Attributes of the Switch Information Attribute Set

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*SwitchType*	enum8	0x00 to 0x01	R	-	M

3.9.2.2.2 *SwitchType* Attribute

The *SwitchType* attribute specifies the basic functionality of the On/Off switching device. This attribute SHALL be set to one of the nonreserved values listed in Table 3-52.

Table 3-52. Values of the *SwitchType* Attribute

Attribute Value	Description	Details
0x00	Toggle	A switch with two physical states. An action by the user (e.g., toggling a rocker switch) moves the switch from state 1 to state 2. The switch then remains in that state until another action from the user returns it to state 1.
0x01	Momentary	A switch with two physical states. An action by the user (e.g., pressing a button) moves the switch from state 1 to state 2. When the user ends his action (e.g., releases the button) the switch returns to state 1.
0x02	Multifunction	A switch that behaves differently depending on user input. Under some conditions it MAY send a toggle or in some other conditions a move command. The behavior of the switch is application-specific but the nature of the switch is clear: it is a multifunction switch.

3.9.2.2.3 Switch Settings Attribute Set

The switch settings attribute set contains the attributes summarized in Table 3-53.

Table 3-53. Attributes of the Switch Settings Attribute Set

Identifier	Name	Type	Range	Access	Default	M/O
0x0010	*SwitchActions*	enum8	0x00 to 0x02	RW	0x00	M

3.9.2.2.3.1 *SwitchActions* Attribute

The *SwitchActions* attribute is 8 bits in length and specifies the commands of the On/Off cluster (see 3.8) to be generated when the switch moves between its two states, as detailed in Table 3-54.

Table 3-54. Values of the *SwitchActions* Attribute

Attribute Value	Command Generated When Arriving at State 2 From State 1	Command Generated When Arriving at State 1 From State 2
0x00	On	Off
0x01	Off	On
0x02	Toggle	Toggle

3.9.2.3 Commands

No commands are gernerated or received by the server.

3.9.3 Client

The client has no cluster specific attributes. No cluster specific commands are generated or received by the client.

3.10 Level

3.10.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface for controlling a characteristic of a device that can be set to a level, for example the brightness of a light, the degree of closure of a door, or the power output of a heater.

NOTE: This cluster specification is a base cluster for generic level control. Also, in this document, is the Level Control for Lighting cluster specification, formerly just Level Control. Level Control for Lighting is derived from this cluster specification, and has further requirements for the lighting application. Please see section 3.19 for the Level Control for Lighting.

3.10.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added
2	added *Options* attribute, state change table; ZLO 1.0; Base cluster (no change) CCB 2085 1775 2281 2147

3.10.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	LVL	Type 1 (client to server)

3.10.1.3 Cluster Identifiers

Derived cluster specifications are defined elsewhere. This base cluster specification MAY be used for generic level control; however, it is recommended to derive another cluster to better define the application and domain requirements. If one of more derived cluster identifiers and the base identifier exists on a device endpoint, then they SHALL all represent a single instance of the device level control. See Chapter 2 – Instance Model for more information.

Identifier	Hierarchy	Name
0x0008	Base	Level (this cluster specificiation)
0x0008	Derived	Level Control for Lighting (3.19)
0x001c	Derived	Pulse Width Modulation (3.20)

3.10.2 Server

3.10.2.1 Dependencies

For many applications, a close relationship between this cluster and the On/Off cluster is needed. This section describes the dependencies that are required when an endpoint that implements this server cluster and also implements the On/Off server cluster.

The *OnOff* attribute of the On/Off cluster and the *CurrentLevel* attribute of the Level Control cluster are intrinsically independent variables, as they are on different clusters. However, when both clusters are implemented on the same endpoint, dependencies MAY be introduced between them. Facilities are provided to introduce dependencies if required.

3.10.2.1.1 Effect of On/Off Commands on the *CurrentLevel* Attribute

The attribute *OnLevel* (see 3.10.2.2.10) determines whether commands of the On/Off cluster have a permanent effect on the *CurrentLevel* attribute or not. If this attribute is defined (i.e., implemented and not 0xff) they do have a permanent effect, otherwise they do not. There is always a temporary effect, due to fading up / down.

The effect on the Level Control cluster on receipt of the various commands of the On/Off cluster are as detailed in Table 3-55. In this table, and throughout this cluster specification, 'level' means the value of the *CurrentLevel* attribute.

Table 3-55. Actions on Receipt for On/Off Commands, when Associated with Level Control

Command	Action On Receipt
On	Temporarily store *CurrentLevel*. Set *CurrentLevel* to the minimum level allowed for the device. Change[16] *CurrentLevel* to *OnLevel*, or to the stored level if *OnLevel* is not defined, over the time period *OnOffTransitionTime*.
Off	Temporarily store *CurrentLevel*. Change *CurrentLevel* to the minimum level allowed for the device over the time period *OnOffTransitionTime*. If *OnLevel* is not defined, set the *CurrentLevel* to the stored level.
Toggle	If the *OnOff* attribute has the value Off, proceed as for the On command. Otherwise proceed as for the Off command.

[16] CCB 2085

Intention of the actions described in the table above is that *CurrentLevel*, which was in effect before any of the On, Off or Toggle commands were issued, shall be restored, after the transition is completed. If another of these commands is received, before the transition is completed, the originally stored *CurrentLevel* shall be preserved and restored[17].

3.10.2.1.2 Effect of Level Control Commands on the *OnOff* Attribute

There are two sets of commands provided in the Level Control cluster. These are identical, except that the first set (Move to Level, Move and Step) SHALL NOT affect the *OnOff* attribute, whereas the second set ('with On/Off' variants) SHALL.

The first set is used to maintain independence between the *CurrentLevel* and *OnOff* attributes, so changing *CurrentLevel* has no effect on the *OnOff* attribute. As examples, this represents the behavior of a volume control with a mute button, or a 'turn to set level and press to turn on/off' light dimmer.

The second set is used to link the *CurrentLevel* and *OnOff* attributes. When the level is reduced to its minimum the *OnOff* attribute is automatically turned to Off, and when the level is increased above its minimum the *OnOff* attribute is automatically turned to On. As an example, this represents the behavior of a light dimmer with no independent on/off switch.

3.10.2.1.3 GlobalSceneControl and Commands with On/Off

If a *Move to Level (with On/off)*, *Move (with on/Off)* or *Step (with On/Off)* command is received that causes a change to the value of the *OnOff* attribute of the On/Off cluster, the value of the *GlobalSceneControl* attribute of the On/Off cluster SHALL be updated according to section 3.8.2.2.2.

3.10.2.2 Attributes

The attributes of the Level Control server cluster are summarized in Table 3-56.

Table 3-56. Attributes of the Level Control Server Cluster

Id	Name	Type	Range	Acc	Default	M/O
0x0000	*CurrentLevel*	uint8	*MinLevel* to *MaxLevel*	RPS	0xff	M
0x0001	*RemainingTime*	uint16	0x0000 to 0xffff	R	0x0000	O
0x0002	*MinLevel*[18]	uint8	0x00 to *MaxLevel*	R	0x00	O
0x0003	*MaxLevel*[19]	uint8	*MinLevel* to 0xff	R	0xff	O
0x0004	*CurrentFrequency*[20]	uint16	*MinFrequency* to *MaxFrequency*	RPS	0x0000	O
0x0005	*MinFrequency*[21]	uint16	0x0000 to *MaxFrequency*	R	0x0000	O
0x0006	*MaxFrequency*[22]	uint16	*MinFrequency* to 0xffff	R	0x0000	O
0x0010	*OnOffTransitionTime*	uint16	0x0000 to 0xffff	RW	0x0000	O

[17] CCB 1775
[18] NFR Quality of Goods
[19] NFR Quality of Goods
[20] NFR Quality of Goods
[21] NFR Quality of Goods
[22] NFR Quality of Goods

Id	Name	Type	Range	Acc	Default	M/O
0x0011	*OnLevel*	uint8	*MinLevel* to *MaxLevel*	RW	0xff	O
0x0012	*OnTransitionTime*	uint16	0x0000 to 0xfffe	RW	0xffff	O
0x0013	*OffTransitionTime*	uint16	0x0000 to 0xfffe	RW	0xffff	O
0x0014	*DefaultMoveRate*	uint16	0x00 to 0xfe	RW	*MS*	O
0x000F	*Options*[23]	map8		RW	0x00	O
0x4000	*StartUpCurrentLevel*[24]	uint8	0x00 to 0xff	RW	*MS*	O

3.10.2.2.1 *CurrentLevel* Attribute

The *CurrentLevel* attribute represents the current level of this device. The meaning of 'level' is device dependent.

3.10.2.2.2 *RemainingTime* Attribute

The *RemainingTime* attribute represents the time remaining until the current command is complete - it is specified in 1/10ths of a second.

3.10.2.2.3 *MinLevel* Attribute

The *MinLevel* attribute indicates the minimum value of *CurrentLevel* that is capable of being assigned.

3.10.2.2.4 *MaxLevel* Attribute

The *MaxLevel* attribute indicates the maximum value of *CurrentLevel* that is capable of being assigned.

3.10.2.2.5 *CurrentFrequency* Attribute

The *CurrentFrequency* attribute represents the frequency that the devices is at *CurrentLevel*. A *CurrentFrequency* of 0 is unknown.

3.10.2.2.6 *MinFrequency* Attribute

The *MinFrequency* attribute indicates the minimum value of *CurrentFrequency* that is capable of being assigned. *MinFrequency* shall be less than or equal to *MaxFrequency*. A value of 0 indicates undefined.

3.10.2.2.7 *MaxFrequency* Attribute

The *MaxFrequency* attribute indicates the maximum value of *CurrentFrequency* that is capable of being assigned. *MaxFrequency* shall be greater than or equal to *MinFrequency*. A value of 0 indicates undefined.

[23] CCB 2085
[24] ZLO 1.0

3.10.2.2.8 *Options* Attribute[25]

The *Options* attribute is meant to be changed only during commissioning. The *Options* attribute is a bitmap that determines the default behavior of some cluster commands. Each command that is dependent on the *Options* attribute SHALL first construct a temporary Options bitmap that is in effect during the command processing. The temporary Options bitmap has the same format and meaning as the *Options* attribute, but includes any bits that may be overridden by command fields.

Below is the format and description of the *Options* attribute and temporary Options bitmap and the effect on dependent commands.

Table 3-57. *Options* Attribute

Bit	Name	Values & Summary
0	ExecuteIfOff	0 – Do not execute command if OnOff is 0x00 (FALSE) 1 – Execute command if OnOff is 0x00 (FALSE)
1	*Reserved for Derived Clusters*	This bit has been defined in these derived clusters for a specific application: Level Control for Lighting

3.10.2.2.8.1 ExecuteIfOff Options Bit

Command execution SHALL NOT continue beyond the *Options* processing if all of these criteria are true:

- The command is one of the 'without On/Off' commands: Move, Move to Level, Stop, or Step.
- The On/Off cluster exists on the same endpoint as this cluster.
- The *OnOff* attribute of the On/Off cluster, on this endpoint, is 0x00 (FALSE).
- The value of the ExecuteIfOff bit is 0.

3.10.2.2.9 *OnOffTransitionTime* Attribute

The *OnOffTransitionTime* attribute represents the time taken to move to or from the target level when On of Off commands are received by an On/Off cluster on the same endpoint. It is specified in 1/10ths of a second.

The actual time taken SHOULD be as close to *OnOffTransitionTime* as the device is able. N.B. If the device is not able to move at a variable rate, the *OnOffTransitionTime* attribute SHOULD NOT be implemented.

3.10.2.2.10 *OnLevel* Attribute

The *OnLevel* attribute determines the value that the *CurrentLevel* attribute is set to when the *OnOff* attribute of an On/Off cluster on the same endpoint is set to On, as a result of processing an On/Off cluster command[26]. If the *OnLevel* attribute is not implemented, or is set to 0xff, it has no effect. For more details see 3.10.2.1.1.

[25] CCB 2085
[26] CCB 2281 "as a result of processing an On/Off cluster command"

3.10.2.2.11 OnTransitionTime Attribute

The *OnTransitionTime* attribute represents the time taken to move the current level from the minimum level to the maximum level when an On command is received by an On/Off cluster on the same endpoint. It is specified in 10ths of a second. If this command is not implemented, or contains a value of 0xffff, the *On/OffTransitionTime* will be used instead.

3.10.2.2.12 OffTransitionTime Attribute

The *OffTransitionTime* attribute represents the time taken to move the current level from the maximum level to the minimum level when an Off command is received by an On/Off cluster on the same endpoint. It is specified in 10ths of a second. If this command is not implemented, or contains a value of 0xffff, the *On/OffTransitionTime* will be used instead.

3.10.2.2.13 DefaultMoveRate Attribute

The *DefaultMoveRate* attribute determines the movement rate, in units per second, when a Move command is received with a Rate parameter of 0xFF.

3.10.2.2.14 StartUpCurrentLevel Attribute

The *StartUpCurrentLevel* attribute SHALL define the desired startup level for a device when it is supplied with power and this level SHALL be reflected in the *CurrentLevel* attribute. The values of the *StartUpCurrentLevel* attribute are listed below:

Table 3-58. Values of the *StartUpCurrentLevel* attribute

Value	Action on power up
0x00	Set the *CurrentLevel* attribute to the minimum value permitted on the device
0xff	Set the *CurrentLevel* attribute to its previous value
other values	Set the *CurrentLevel* attribute to this value

3.10.2.3 Commands Received

The command IDs for the Level Control cluster are listed below.

Table 3-59. Command IDs for the Level Control Cluster

ID	Description	M/O
0x00	Move to Level	M
0x01	Move	M
0x02	Step	M
0x03	Stop	M
0x04	Move to Level (with On/Off)	M

ID	Description	M/O
0x05	Move (with On/Off)	M
0x06	Step (with On/Off)	M
0x07	Stop	M
0x08	Move to Closest Frequency[27]	M:*CurrentFrequency* attribute supported

3.10.2.3.1 Move to Level Command

3.10.2.3.1.1 Payload Format

The Move to Level command payload SHALL be formatted as illustrated in Figure 3-40.

Figure 3-40. Format of the Move to Level Command Payload

Octets	1	2	0/1	0/1
Data Type	uint8	uint16	map8	map8
Field Name	Level	Transition time	OptionsMask[28]	OptionsOverride[29]

3.10.2.3.1.2 Effect on Receipt

The OptionsMask & OptionsOverride fields SHALL both be present or both omitted in the command. A temporary Options bitmap SHALL be created from the *Options* attribute, using the OptionsMask & OptionsOverride fields, if present. Each bit of the temporary Options bitmap SHALL be determined as follows:

Each bit in the *Options* attribute SHALL determine the corresponding bit in the temporary Options bitmap, unless the OptionsMask field is present and has the corresponding bit set to 1, in which case the corresponding bit in the OptionsOverride field SHALL determine the corresponding bit in the temporary Options bitmap.

The resulting temporary Options bitmap SHALL then be processed as defined in section 3.10.2.2.3.

On receipt of this command, a device SHALL move from its current level to the value given in the Level field. The meaning of 'level' is device dependent – e.g., for a light it MAY mean brightness level.

The movement SHALL be as continuous as technically practical, i.e., not a step function, and the time taken to move to the new level SHALL be equal to the value of the Transition time field, in tenths of a second, or as close to this as the device is able.

If the Transition time field takes the value 0xffff then the time taken to move to the new level SHALL instead be determined by the *OnOffTransitionTime* attribute. If *OnOffTransitionTime*, which is an optional attribute, is not present, the device SHALL move to its new level as fast as it is able.

If the device is not able to move at a variable rate, the Transition time field MAY be disregarded.

3.10.2.3.2 Move Command

3.10.2.3.2.1 Payload Format

[27] NFR Quality of Goods
[28] CCB 2085
[29] CCB 2085

The Move command payload SHALL be formatted as illustrated in Figure 3-41.

Figure 3-41. Format of the Move Command Payload

Octets	1	1	0/1	0/1
Data Type	enum8	uint8	map8	map8
Field Name	Move mode	Rate	OptionsMask[30]	OptionsOverride[31]

3.10.2.3.2.2 Move Mode Field

The Move mode field SHALL be one of the non-reserved values in Table 3-60.

Table 3-60. Values of the Move Mode Field

Fade Mode Value	Description
0x00	Up
0x01	Down

3.10.2.3.2.3 Rate Field

The Rate field specifies the rate of movement in units per second. The actual rate of movement SHOULD be as close to this rate as the device is able. If the Rate field is 0xFF, then the value in *DefaultMoveRate* attribute SHALL be used. If the Rate field is 0xFF and the *DefaultMoveRate* attribute is not supported, then the device SHOULD move as fast as it is able. If the device is not able to move at a variable rate, this field MAY be disregarded.

3.10.2.3.2.4 Effect on Receipt

On receipt of this command, a device SHALL first create and process a temporary Options bitmap as described in section 3.10.2.3.1.2.

On receipt of this command, a device SHALL move from its current level in an up or down direction in a continuous fashion, as detailed in Table 3-61.

Table 3-61. Actions on Receipt for Move Command

Fade Mode	Action on Receipt
Up	Increase the device's level at the rate given in the Rate field. If the level reaches the maximum allowed for the device, stop.
Down	Decrease the device's level at the rate given in the Rate field. If the level reaches the minimum allowed for the device, stop.

3.10.2.3.3 Step Command

3.10.2.3.3.1 Payload Format

[30] CCB 2085
[31] CCB 2085

The Step command payload SHALL be formatted as illustrated in Figure 3-42.

Figure 3-42. Format of the Step Command Payload

Octets	1	1	2	0/1	0/1
Data Type	enum8	uint8	uint16	map8	map8
Field Name	Step mode	Step size	Transition time	OptionsMask[32]	OptionsOverride[33]

The Step mode field SHALL be one of the non-reserved values in Table 3-62.

Table 3-62. Values of the Step Mode Field

Fade Mode Value	Description
0x00	Up
0x01	Down

The Transition time field specifies the time that SHALL be taken to perform the step, in tenths of a second. A step is a change in the *CurrentLevel* of 'Step size' units. The actual time taken SHOULD be as close to this as the device is able. If the Transition time field is 0xffff the device SHOULD move as fast as it is able.

If the device is not able to move at a variable rate, the Transition time field MAY be disregarded.

3.10.2.3.3.2 Effect on Receipt

On receipt of this command, a device SHALL first create and process a temporary Options bitmap as described in section 3.10.2.3.1.2.

On receipt of this command, a device SHALL move from its current level in an up or down direction as detailed in Table 3-63.

Table 3-63. Actions on Receipt for Step Command

Fade Mode	Action on Receipt
Up	Increase *CurrentLevel* by 'Step size' units, or until it reaches the maximum level allowed for the device if this reached in the process. In the latter case, the transition time SHALL be proportionally reduced.
Down	Decrease *CurrentLevel* by 'Step size' units, or until it reaches the minimum level allowed for the device if this reached in the process. In the latter case, the transition time SHALL be proportionally reduced.

3.10.2.3.4 Stop Command

3.10.2.3.4.1 Payload Format

[32] CCB 2085
[33] CCB 2085

The command payload SHALL be formatted as illustrated below.

Figure 3-43. Format of the Command Payload

Octets	0/1	0/1
Data Type	map8	map8
Field Name	OptionsMask[34]	OptionsOverride[35]

3.10.2.3.4.2 Effect of Receipt

On receipt of this command, a device SHALL first create and process a temporary Options bitmap as described in section 3.10.2.3.1.2.

Upon receipt of this command, any Move to Level, Move or Step command (and their 'with On/Off' variants) currently in process SHALL be terminated. The value of *CurrentLevel* SHALL be left at its value upon receipt of the Stop command, and *RemainingTime* SHALL be set to zero.

This command has two entries in Table 3-5, one for the Move to Level, Move and Set commands, and one for their 'with On/Off' counterparts. This is solely for symmetry, to allow easy choice of one or other set of commands – the Stop commands are identical.

3.10.2.3.5 Move to Closest Frequency Command

This command shall be mandatory if the CurrentFrequency attribute is supported.

3.10.2.3.5.1 Payload Format

The command payload SHALL be formatted as illustrated below.

Figure 3-44. Format of the Command Payload

Octets	1
Data Type	uint16
Field Name	Frequency

3.10.2.3.5.2 Effect of Receipt

Upon receipt of this command, the device shall change its current frequency to the requested frequency, or to the closest frequency that it can generate. If the device cannot approximate the frequency, then it shall return a default response with an error code of INVALID_VALUE. Determining if a requested frequency can be approximated by a supported frequency is a manufacturer-specific decision.

[34] CCB 2085
[35] CCB 2085

3.10.2.3.6 'With On/Off' Commands

The Move to Level (with On/Off), Move (with On/Off) and Step (with On/Off) commands have identical payloads to the Move to Level, Move and Step commands respectively, except for the OptionsMask and OptionsOverride fields. They also have the same effects, except for the following additions.

Before commencing any command that has the effect of setting the *CurrentLevel* above the minimum level allowed by the device[36], the OnOff attribute of the On/Off cluster on the same endpoint, if implemented, SHALL be set to On.

If any command that has the effect of setting the[37] CurrentLevel to the minimum level allowed by the device, the OnOff attribute of the On/Off cluster on the same endpoint, if implemented, SHALL be set to Off.

3.10.2.4 Commands Generated

The server generates no commands.

3.10.3 Client

The client has no cluster specific attributes. The client generates the cluster specific commands received by the server (see 3.10.2.4), as required by the application. No cluster specific commands are received by the client.

3.11 Alarms

3.11.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

Attributes and commands for sending alarm notifications and configuring alarm functionality.

Alarm conditions and their respective alarm codes are described in individual clusters, along with an alarm mask field. Alarm notifications are reported to subscribed targets using binding.

Where an alarm table is implemented, all alarms, masked or otherwise, are recorded and MAY be retrieved on demand.

Alarms MAY either reset automatically when the conditions that cause are no longer active, or MAY need to be explicitly reset.

3.11.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added

3.11.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	ALM	Type 2 (server to client)

[36] CCB 2147 clarify that CurrentLevel letting effects On/Off, not increasing or decreasing
[37] CCB 2147 clarify that CurrentLevel letting effects On/Off, not increasing or decreasing

3.11.1.3 Cluster Identifiers

Identifier	Name
0x0009	Alarms

3.11.2 Server

3.11.2.1 Dependencies

Any endpoint which implements time stamping SHALL also implement the Time server cluster.

3.11.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 3-64.

Table 3-64. Alarms Cluster Attribute Sets

Attribute Set Identifier	Description
0x000	Alarm Information

3.11.2.2.1 Alarm Information Attribute Set

The Alarm Information attribute set contains the attributes summarized in Table 3-65.

Table 3-65. Attributes of the Alarm Information Attribute Set

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*AlarmCount*	uint16	0x00 to maximum defined in profile	R	0x00	O

3.11.2.2.1.1 *AlarmCount* Attribute

The *AlarmCount* attribute is 16 bits in length and specifies the number of entries currently in the alarm table. This attribute SHALL be specified in the range 0x00 to the maximum defined in the profile using this cluster.

If alarm logging is not implemented this attribute SHALL always take the value 0x00.

3.11.2.3 Alarm Table

The alarm table is used to store details of alarms generated within the devices. Alarms are requested by clusters which have alarm functionality, e.g., when attributes take on values that are outside ‘safe’ ranges.

The maximum number of entries in the table is device dependent.

When an alarm is generated, a corresponding entry is placed in the table. If the table is full, the earliest entry is replaced by the new entry.

Once an alarm condition has been reported the corresponding entry in the table is removed.

3.11.2.3.1 Alarm Table Format

The format of an alarm table entry is illustrated in Table 3-66Format of the Alarm Table.

Table 3-66. Format of the Alarm Table

Field	Type	Valid Range	Description
Alarm code	enum8	0x00 to 0xff	Identifying code for the cause of the alarm, as given in the specification of the cluster whose attribute generated this alarm.
Cluster identifier	clusterId	0x0000 to 0xffff	The identifier of the cluster whose attribute generated this alarm.
Time stamp	uint32	0x00000000 to 0xffffffff	The time at which the alarm occurred or 0xffffffff if no time information is available. This time is taken from a Time server cluster, which must be present on the same endpoint.

3.11.2.4 Commands Received

The received command IDs for the Alarms cluster are listed in Table 3-67.

Table 3-67. Received Command IDs for the Alarms Cluster

Command Identifier Field Value	Description	M/O
0x00	Reset Alarm	M
0x01	Reset all alarms	M
0x02	Get Alarm	O
0x03	Reset alarm log	O

3.11.2.4.1 Reset Alarm Command

This command resets a specific alarm. This is needed for some alarms that do not reset automatically. If the alarm condition being reset was in fact still active then a new notification will be generated and, where implemented, a new record added to the alarm log.

3.11.2.4.1.1 Payload Format

The Reset Alarm command payload SHALL be formatted as illustrated in Figure 3-45.

Figure 3-45. Format of the Reset Alarm Command Payload

Octets	1	2
Data Type	enum8	clusterId
Field Name	Alarm code	Cluster identifier

3.11.2.4.2 Reset All Alarms Command

This command resets all alarms. Any alarm conditions that were in fact still active will cause a new notification to be generated and, where implemented, a new record added to the alarm log.

3.11.2.4.3 Get Alarm Command

This command causes the alarm with the earliest generated alarm entry in the alarm table to be reported in a get alarm response command 3.11.2.5.2. This command enables the reading of logged alarm conditions from the alarm table. Once an alarm condition has been reported the corresponding entry in the table is removed.

This command does not have a payload.

3.11.2.4.4 Reset Alarm Log Command

This command causes the alarm table to be cleared, and does not have a payload.

3.11.2.5 Commands Generated

The generated command IDs for the Alarms cluster are listed in Table 3-68.

Table 3-68. Generated Command IDs for the Alarms Cluster

Command Identifier Field Value	**Description**	**M/O**
0x00	Alarm	M
0x01	Get alarm response	O

3.11.2.5.1 Alarm Command

The alarm command signals an alarm situation on the sending device.

An alarm command is generated when a cluster which has alarm functionality detects an alarm condition, e.g., an attribute has taken on a value that is outside a ‘safe’ range. The details are given by individual cluster specifications.

3.11.2.5.1.1 Payload Format

The alarm command payload SHALL be formatted as illustrated in Figure 3-46.

Figure 3-46. Format of the Alarm Command Payload

Octets	1	2
Data Type	enum8	clusterId
Field Name	Alarm code	Cluster identifier

3.11.2.5.2 Get Alarm Response Command

The get alarm response command returns the results of a request to retrieve information from the alarm log, along with a time stamp indicating when the alarm situation was detected.

3.11.2.5.2.1 Payload Format

The get alarm response command payload SHALL be formatted as illustrated in Figure 3-47.

Figure 3-47. Format of the Get Alarm Response Command Payload

Octets	1	0/1	0/2	0/4
Data Type	enum8	enum8	clusterId	uint32
Field Name	Status	Alarm code	Cluster identifier	Time stamp

If there is at least one alarm record in the alarm table then the status field is set to SUCCESS. The alarm code, cluster identifier and time stamp fields SHALL all be present and SHALL take their values from the item in the alarm table that they are reporting.

If there are no more alarms logged in the alarm table then the status field is set to NOT_FOUND and the alarm code, cluster identifier and time stamp fields SHALL be omitted.

3.11.3 Client

The client has no cluster specific attributes. The client generates the cluster specific commands received by the server (see 3.11.2.4), as required by the application. The client receives the cluster specific commands generated by the server (see 3.11.2.5).

3.12 Time

3.12.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides a basic interface to a real-time clock. The clock time MAY be read and also written, in order to synchronize the clock (as close as practical) to a time standard. This time standard is the number of seconds since 0 hrs 0 mins 0 sec on 1st January 2000 UTC (Universal Coordinated Time).

The cluster also includes basic functionality for local time zone and daylight saving time.

4908 #### 3.12.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added

4909 #### 3.12.1.2 Classification

Hierarchy	Role	PICS Code
Base	Application	T

4910 #### 3.12.1.3 Cluster Identifiers

Identifier	Name
0x000a	Time

4911 ### 3.12.2 Server

4912 #### 3.12.2.1 Attributes

4913 The server supports the attributes shown in Table 3-69.

4914 **Table 3-69. Attributes of the Time Server Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*Time*	UTC	0x00000000 to 0xfffffffe	RW	0xffffffff	M
0x0001	*TimeStatus*	map8	0000 xxxx	RW	0b00000000	M
0x0002	*TimeZone*	int32	-86400 to +86400	RW	0x00000000	O
0x0003	*DstStart*	uint32	0x00000000 to 0xfffffffe	RW	0xffffffff	O
0x0004	*DstEnd*	uint32	0x00000000 to 0xfffffffe	RW	0xffffffff	O
0x0005	*DstShift*	int32	-86400 to +86400	RW	0x00000000	O
0x0006	*StandardTime*	uint32	0x00000000 to 0xfffffffe	R	0xffffffff	O
0x0007	*LocalTime*	uint32	0x00000000 to 0xfffffffe	R	0xffffffff	O
0x0008	*LastSetTime*	UTC	0x00000000 to 0xffffffff	R	0xffffffff	O
0x0009	*ValidUntilTime*	UTC	0x00000000 to 0xffffffff	RW	0xffffffff	O

3.12.2.1.1 *Time* Attribute

The *Time* attribute is 32 bits in length and holds the time value of a real time clock. This attribute has data type UTCTime, but note that it MAY not actually be synchronized to UTC - see discussion of the *TimeStatus* attribute.

If the Master bit of the *TimeStatus* attribute has a value of 0, writing to this attribute SHALL set the real time clock to the written value, otherwise it cannot be written. The value 0xffffffff indicates an invalid time.

3.12.2.1.2 *TimeStatus* Attribute

The *TimeStatus* attribute holds a number of bit fields, as detailed in Table 3-70.

Table 3-70. Bit Values of the *TimeStatus* Attribute

Attribute Bit Number	Meaning	Values
0	Master	1 – master clock 0 – not master clock
1	Synchronized	1 – synchronized 0 – not synchronized
2	MasterZoneDst	1 – master for Time Zone and DST 0 – not master for Time Zone and DST
3	Superseding	1 – time synchronization SHOULD be superseded 0 – time synchronization SHOULD not be superseded

The Master and Synchronized bits together provide information on how closely the *Time* attribute conforms to the time standard.

The Master bit specifies whether the real time clock corresponding to the *Time* attribute is internally set to the time standard. This bit is not writeable – if a value is written to the *TimeStatus* attribute, this bit does not change.

The Synchronized bit specifies whether *Time* has been set over the network to synchronize it (as close as MAY be practical) to the time standard (see 3.12.1). This bit must be explicitly written to indicate this – i.e., it is not set automatically on writing to the *Time* attribute. If the Master bit is 1, the value of this bit is 0.

If both the Master and Synchronized bits are 0, the real time clock has no defined relationship to the time standard (e.g., it MAY record the number of seconds since the device was initialized).

The MasterZoneDst bit specifies whether the *TimeZone, DstStart, DstEnd* and *DstShift* attributes are set internally to correct values for the location of the clock. If not, these attributes need to be set over the network. This bit is not writeable – if a value is written to the *TimeStatus* attribute, this bit does not change.

Devices SHALL synchronize to a Time server with the highest rank according to the following rules, listed in order of precedence:

- A server with the Superseding bit set SHALL be chosen over a server without the bit set.
- A server with the Master bit SHALL be chosen over a server without the bit set.

- The server with the lower short address SHALL be chosen (note that this means a coordinator with the Superseding and Master bit set will always be chosen as the network time server).
- A Time server with neither the Master nor Synchronized bits set SHOULD not be chosen as the network time server.

3.12.2.1.3 *TimeZone* Attribute

The *TimeZone* attribute indicates the local time zone, as a signed offset in seconds from the *Time* attribute value. The value 0xffffffff indicates an invalid time zone.

The local Standard Time, i.e., the time adjusted for the time zone, but not adjusted for Daylight Saving Time (DST) is given by

Standard Time = *Time* + *TimeZone*

The range of this attribute is +/- one day. Note that the actual range of physical time zones on the globe is much smaller than this, so the manufacturer has the option to impose a smaller range.

If the MasterZoneDst bit of the *TimeStatus* attribute has a value of 1, this attribute cannot be written.

3.12.2.1.4 *DstStart* Attribute

The *DstStart* attribute indicates the DST start time in seconds. The value 0xffffffff indicates an invalid DST start time. For semantic purposes *DstStart* and *DstEnd* are actually type UTCTime.

The Local Time, i.e., the time adjusted for both the time zone and DST, is given by

Local Time = Standard Time + *DstShift* (if *DstStart* <= *Time* <= *DstEnd*)

Local Time = Standard Time (if *Time* < *DstStart* or *Time* > *DstEnd*)

Note that the three attributes DstStart, DstEnd and DstShift are optional, but if any one of them is implemented the other two must also be implemented.

Note that this attribute SHOULD be set to a new value once every year.

If the MasterZoneDst bit of the *TimeStatus* attribute has a value of 1, this attribute cannot be written.

3.12.2.1.5 *DstEnd* Attribute

The *DstEnd* attribute indicates the DST end time in seconds. The value 0xffffffff indicates an invalid DST end time. For semantic purposes *DstStart* and *DstEnd* are actually type UTCTime.

Note that this attribute SHOULD be set to a new value once every year, and SHOULD be written synchronously with the *DstStart* attribute.

If the MasterZoneDst bit of the *TimeStatus* attribute has a value of 1, this attribute cannot be written.

3.12.2.1.6 *DstShift* Attribute

The *DstShift* attribute represents a signed offset in seconds from the standard time, to be applied between the times *DstStart* and *DstEnd* to calculate the Local Time (see 3.12.2.1.4). The value 0xffffffff indicates an invalid DST shift.

The range of this attribute is +/- one day. Note that the actual range of DST values employed by countries is much smaller than this, so the manufacturer has the option to impose a smaller range.

If the MasterZoneDst bit of the *TimeStatus* attribute has a value of 1, this attribute cannot be written.

3.12.2.1.7 *StandardTime* Attribute

The local Standard Time is given by the equation in 3.12.2.1.3. Another device on the network MAY calculate this time by reading the *Time* and *TimeZone* attributes and adding them together. If implemented however, the optional StandardTime attribute indicates this time directly. The value 0xffffffff indicates an invalid Standard Time.

3.12.2.1.8 *LocalTime* Attribute

The Local Time is given by the equation in 3.12.2.1.4. Another device on the network MAY calculate this time by reading the *Time*, *TimeZone*, *DstStart*, *DstEnd* and *DstShift* attributes and performing the calculation. If implemented however, the optional LocalTime attribute indicates this time directly. The value 0xffffffff indicates an invalid Local Time.

3.12.2.1.9 *LastSetTime* Attribute

The LastSetTime attribute indicates the most recent time that the *Time* attribute was set, either internally or over the network (thus it holds a copy of the last value that *Time* was set to). This attribute is set automatically, so is Read Only. The value 0xffffffff indicates an invalid LastSetTime.

3.12.2.1.10 *ValidUntilTime* Attribute

The ValidUntilTime attribute indicates a time, later than LastSetTime, up to which the Time attribute MAY be trusted. 'Trusted' means that the difference between the Time attribute and the true UTC time is less than an acceptable error. The acceptable error is not defined by this cluster specification, but MAY be defined by the application profile in which devices that use this cluster are specified.

Note: The value that the ValidUntilTime attribute SHOULD be set to depends both on the acceptable error and the drift characteristics of the real time clock in the device that implements this cluster, which must therefore be known by the application entity that sets this value.

The value 0xffffffff indicates an invalid ValidUntilTime.

3.12.2.2 Commands Received

The server receives no commands except those to read and write attributes.

3.12.2.3 Commands Generated

The server generates no cluster specific commands.

3.12.3 Client

The client has no cluster specific attributes. No cluster specific commands are generated or received by the client.

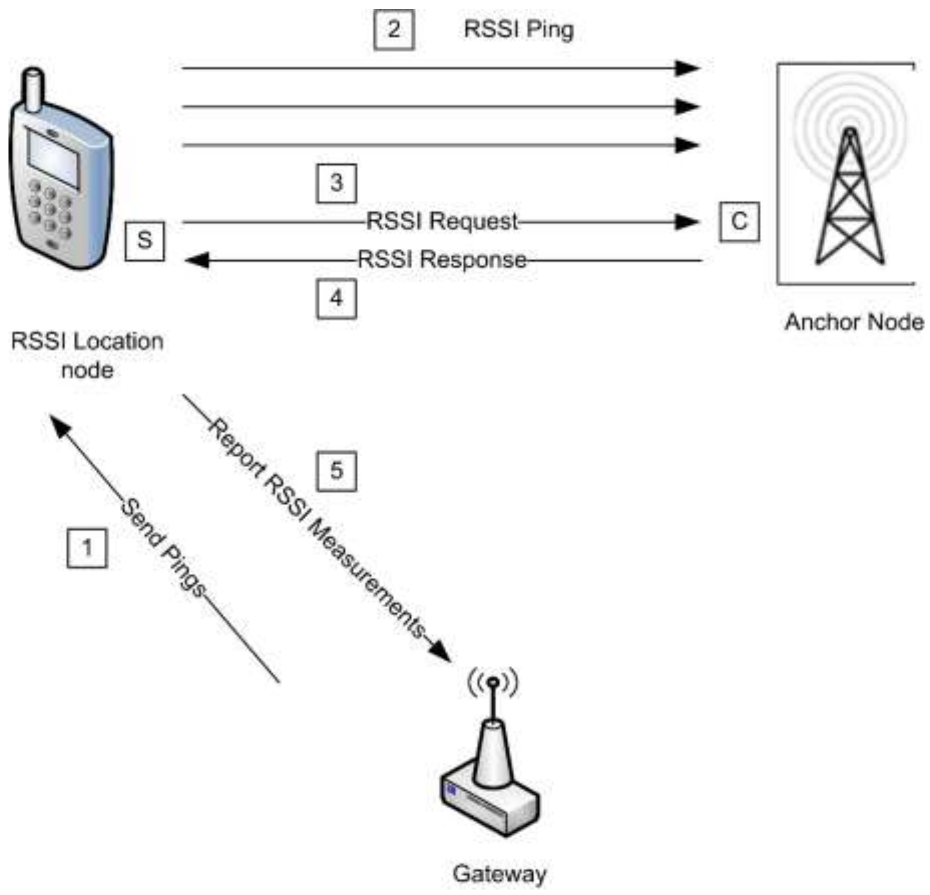
3.13 RSSI Location

3.13.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides a means for exchanging Received Signal Strength Indication (RSSI) information among one hop devices as well as messages to report RSSI data to a centralized device that collects all the RSSI data in the network. An example of the usage of RSSI location cluster is shown in Figure 3-48.

5010 **Figure 3-48. Example of Usage of RSSI Location Cluster**



5011

5012 3.13.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added

5013 3.13.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	RSSI

5014 3.13.1.3 Cluster Identifiers

Identifier	Name
0x000b	RSSI

3.13.2 Server

3.13.2.1 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 3-71.

Table 3-71. Location Attribute Sets

Attribute Set Identifier	Description
0x000	Location Information
0x001	Location Settings

3.13.2.1.1 Location Information Attribute Set

The Location Information attribute set contains the attributes summarized in Table 3-72.

Table 3-72. Attributes of the Location Information Attribute Set

Identifier	Name	Type	Range	Access	Def	M/O
0x0000	*LocationType*	data8	0000xxxx	RW	-	M
0x0001	*LocationMethod*	enum8	0x00 to 0xff	RW	-	M
0x0002	*LocationAge*	uint16	0x0000 to 0xffff	R	-	O
0x0003	*QualityMeasure*	uint8	0x00 to 0x64	R	-	O
0x0004	*NumberOfDevices*	uint8	0x00 to 0xff	R	-	O

3.13.2.1.1.1 *LocationType* Attribute

The *LocationType* attribute is 8 bits long and is divided into bit fields. The meanings of the individual bit fields are detailed in Table 3-73.

Table 3-73. Bit Values of the *LocationType* Attribute

Bit Field (Bit Numbers)	Meaning	Values
0	Absolute	1 – Absolute location 0 – Measured location
1	2-D	1 – Two dimensional 0 – Three dimensional
2-3	Coordinate System	0 – Rectangular (installation-specific origin and orientation)

The Absolute bit field indicates whether the location is a known absolute location or is calculated.

The 2-D bit field indicates whether the location information is two- or three-dimensional. If the location information is two-dimensional, Coordinate 3 is unknown and SHALL be set to 0x8000.

The Coordinate System bit field indicates the geometry of the system used to express the location coordinates. If the field is set to zero, the location coordinates are expressed using the rectangular coordinate system. All other values are reserved.

3.13.2.1.1.2 *LocationMethod* Attribute

The *LocationMethod* attribute SHALL be set to one of the non-reserved values in Table 3-74.

Table 3-74. Values of the *LocationMethod* Attribute

Value	Method	Description
0x00	Lateration	A method based on RSSI measurements from three or more sources.
0x01	Signposting	The location reported is the location of the neighboring device with the strongest received signal.
0x02	RF fingerprinting	RSSI signatures are collected into a database at commissioning time. The location reported is the location taken from the RSSI signature database that most closely matches the device’s own RSSI signature.
0x03	Out of band	The location is obtained by accessing an out-of-band device (that is, the device providing the location is not part of the network).
0x04	Centralized	The location is performed in a centralized way (e.g., by the GW) by a device on the network. Different from the above because the device performing the localization is part of the network.
0x40 to 0xff	-	Reserved for manufacturer specific location methods.

3.13.2.1.1.3 *LocationAge* Attribute

The *LocationAge* attribute indicates the amount of time, measured in seconds, that has transpired since the location information was last calculated. This attribute is not valid if the Absolute bit of the *LocationType* attribute is set to one.

3.13.2.1.1.4 *QualityMeasure* Attribute

The *QualityMeasure* attribute is a measure of confidence in the corresponding location information. The higher the value, the more confident the transmitting device is in the location information. A value of 0x64 indicates complete (100%) confidence and a value of 0x00 indicates zero confidence. (Note: no fixed confidence metric is mandated – the metric MAY be application and manufacturer dependent.)

This field is not valid if the Absolute bit of the *LocationType* attribute is set to one.

3.13.2.1.1.5 *NumberOfDevices* Attribute

The *NumberOfDevices* attribute is the number of devices whose location data were used to calculate the last location value. This attribute is related to the *QualityMeasure* attribute.

3.13.2.1.2 Location Settings Attribute Set

The Location Settings attribute set contains the attributes summarized in Table 3-75.

Table 3-75. Attributes of the Location Settings Attribute Set

Identifier	Name	Type	Range	Access	Def	M/O
0x0010	*Coordinate1*	int16	0x8000 to 0x7fff	RW	-	M
0x0011	*Coordinate2*	int16	0x8000 to 0x7fff	RW	-	M
0x0012	*Coordinate3*	int16	0x8000 to 0x7fff	RW	-	O
0x0013	*Power*	int16	0x8000 to 0x7fff	RW	-	M
0x0014	*PathLossExponent*	uint16	0x0000 to 0xffff	RW	-	M
0x0015	*ReportingPeriod*	uint16	0x0000 to 0xffff	RW	-	O
0x0016	*CalculationPeriod*	uint16	0x0000 to 0xffff	RW	-	O
0x0017	*NumberRSSIMeasurements*	uint8	0x01 to 0xff	RW	-	M

3.13.2.1.2.1 *Coordinate* 1,2,3 Attributes

The *Coordinate1, Coordinate2 and Coordinate3* attributes are signed 16-bit integers, and represent orthogonal linear coordinates x, y, z in meters as follows.

x = *Coordinate1* / 10, y = *Coordinate2* / 10, z = *Coordinate3* / 10

The range of x is -3276.7 to 3276.7 meters, corresponding to *Coordinate1* between 0x8001 and 0x7fff. The same range applies to y and z. A value of 0x8000 for any of the coordinates indicates that the coordinate is unknown.

3.13.2.1.2.2 *Power* Attribute

The *Power* attribute specifies the value of the average power P_0, measured in dBm, received at a reference distance of one meter from the transmitter.

P_0 = *Power* / 100

A value of 0x8000 indicates that *Power* is unknown.

3.13.2.1.2.3 *PathLossExponent* Attribute

The *PathLossExponent* attribute specifies the value of the Path Loss Exponent n, an exponent that describes the rate at which the signal power decays with increasing distance from the transmitter.

n = *PathLossExponent* / 100

A value of 0xffff indicates that *PathLossExponent* is unknown.

The signal strength in dBm at a distance d meters from the transmitter is given by

$P = P_0 - 10n \times \log_{10}(d)$

where

P is the power in dBm at the receiving device.

P0 is the average power in dBm received at a reference distance of 1meter from the transmitter.

n is the path loss exponent.

d is the distance in meters between the transmitting device and the receiving device.

3.13.2.1.2.4 *ReportingPeriod* Attribute

The *ReportingPeriod* attribute specifies the time in seconds between successive reports of the device's location by means of the Location Data Notification command. The minimum value this attribute can take is specified by the profile in use. If *ReportingPeriod* is zero, the device does not automatically report its location. Note that location information can always be polled at any time.

3.13.2.1.2.5 *CalculationPeriod* Attribute

The *CalculationPeriod* attribute specifies the time in milliseconds between successive calculations of the device's location. If *CalculationPeriod* is less than the physically possible minimum period that the calculation can be performed, the calculation will be repeated as frequently as possible. In case of centralized location (*LocationMethod* attribute equal to Centralized) the *CalculationPeriod* attribute specifies the period between successive RSSI ping commands.

3.13.2.1.2.6 *NumberRSSIMeasurements* Attribute

The *NumberRSSIMeasurements* attribute specifies the number of RSSI measurements to be used to generate one location estimate. The measurements are averaged to improve accuracy. *NumberRSSIMeasurements* must be greater than or equal to 1. In the case of centralized location (*LocationMethod* attribute equal to Centralized) the *NumberRSSIMeasurements* attribute specifies the number of successive RSSI Ping commands to be sent by the server side of location cluster.

3.13.2.2 Commands Received

The received command IDs for the Location cluster are listed in Table 3-76.

Table 3-76. Received Command IDs for the Location Cluster

Command Identifier Field Value	Description	M/O
0x00	Set Absolute Location	M
0x01	Set Device Configuration	M
0x02	Get Device Configuration	M
0x03	Get Location Data	M
0x04	RSSI Response	O
0x05	Send Pings	O
0x06	Anchor Node Announce	O

3.13.2.2.1 Set Absolute Location Command

This command is used to set a device’s absolute (known, not calculated) location and the channel parameters corresponding to that location.

3.13.2.2.1.1 Payload Format

The Set Absolute Location command payload SHALL be formatted as illustrated in Figure 3-49.

Figure 3-49. Format of the Set Absolute Location Command Payload

Octets	2	2	2	2	2
Data Type	int16	int16	int16	int16	uint16
Field Name	Coordinate 1	Coordinate 2	Coordinate 3	Power	Path Loss Exponent

The fields of the payload correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions of the individual attributes.

The three coordinate fields SHALL contain the absolute location (known, not calculated) of the destination device. If any coordinate field(s) is not known, the value(s) SHALL be set to 0x8000.

3.13.2.2.1.2 Effect on Receipt

On receipt of this command, the device SHALL update the attributes corresponding to (i.e., with the same names as) the payload fields.

3.13.2.2.2 Set Device Configuration Command

This command is used to set a device's location parameters, which will be used for calculating and reporting measured location. This command is invalid unless the Absolute bit of the *LocationType* attribute has a value of 0.

3.13.2.2.2.1 Payload Format

The Set Device Configuration command payload SHALL be formatted as illustrated in Figure 3-50.

Figure 3-50. Format of the Set Device Configuration Payload

Octets	2	2	2	1	2
Data Type	int16	uint16	uint16	uint8	uint16
Field Name	Power	Path Loss Exponent	Calculation Period	Number RSSI Measurements	Reporting Period

The fields of the payload correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions of the individual attributes.

3.13.2.2.2.2 Effect on Receipt

On receipt of this command, the device SHALL update the attributes corresponding to (i.e., with the same names as) the payload fields.

3.13.2.2.3 Get Device Configuration Command

This command is used to request the location parameters of a device. The location parameters are used for calculating and reporting measured location.

3.13.2.2.3.1 Payload Format

The Get Device Configuration command payload SHALL be formatted as illustrated in Figure 3-51.

Figure 3-51. Format of the Get Device Configuration Payload

Octets	8
Data Type	EUI64
Field Name	Target Address

The Target Address field contains the 64-bit IEEE address of the device for which the location parameters are being requested. This field MAY contain the address of the sending device, the address of the receiving device or the address of a third device.

Note: one reason a device MAY request its own configuration is that there MAY be a designated device which holds the configurations of other devices for distribution at commissioning time. It is also possible that the device MAY lose its configuration settings for some other reason (loss of power, reset). In the case of a third device, that device MAY sleep a lot and not be easily accessible.

3.13.2.2.3.2 Effect on Receipt

On receipt of this command, the device SHALL generate a Device Configuration Response command (3.13.2.3.1).

3.13.2.2.4 Get Location Data Command

This command is used to request a device's location information and channel parameters. It MAY be sent as a unicast, multicast or broadcast frame. When sent as a broadcast frame, care SHOULD be taken to minimize the risk of a broadcast 'storm' - in particular, it is recommended that the broadcast radius is set to 1.

(**Note:** devices MAY or MAY not acquire and store information on other devices' locations such that this information MAY be requested by another device. This is application dependent.)

3.13.2.2.4.1 Payload Format

The Get Location Data command payload SHALL be formatted as illustrated in Figure 3-52.

Figure 3-52. Format of the Get Location Data Payload

Bits	3	1	1	1	1	1	8	0 / 64
Data Type	map8						uint8	EUI64
Field Name	Reserved	Compact Response	Broadcast Response	Broadcast Indicator	Recalculate	Absolute Only	Number Responses	Target Address

The highest 3 bits of the first octet are reserved and SHALL be set to zero.

The Absolute Only field (bit 0 of the first octet) specifies the type of location information being requested. If the Absolute Only field is set to one, the device is only requesting absolute location information (a device MAY want to gather absolute node locations for use in its own location calculations, and MAY not be interested in neighbors with calculated values). Otherwise, if the field is set to zero, the device is requesting all location information (absolute and calculated).

The Recalculate field (bit 1 of the first octet) indicates whether the device is requesting that a new location calculation be performed. If the field is set to zero, the device is requesting the currently stored location information. Otherwise, if the field is set to one, the device is requesting that a new calculation be performed. This field is only valid if the Absolute Only field is set to zero.

The Broadcast Indicator field (bit 2 of the first octet) indicates whether the command is being sent as a unicast, multicast or broadcast frame. If the field is set to one, the command is sent as a broadcast or multicast, else it is sent as a unicast.

The Broadcast Response field (bit 3 of the first octet) indicates whether subsequent responses after the first (where the Number Responses field is greater than one) SHALL be unicast or broadcast. Broadcast responses can be used as a 'location beacon'.

The Compact Response field (bit 4 of the first octet) indicates whether subsequent responses after the first (where the Number Responses field is greater than one) SHALL be sent using the Location Data Notification or the Compact Location Data Notification command.

The Number Responses field indicates the number of location responses to be returned. The information to be returned is evaluated this number of times, with a period equal to the value of the *ReportingPeriod* attribute, and a separate response is sent for each evaluation. This field SHALL have a minimum value of one. Values greater than one are typically used for situations where locations are changing.

The Target Address field contains the 64-bit IEEE address of the device for which the location information and channel parameters are being requested. If the Broadcast Indicator field is set to zero (i.e., the command is sent as a unicast) this field MAY contain the address of the receiving device, the address of the sending device or the address of any other device. If the Broadcast Indicator field is set to one (i.e., the command is sent as a broadcast or multicast) the target address is implicitly that of the receiving device, so this field SHALL be omitted.

3.13.2.2.4.2 Effect on Receipt

On receipt of this command, if the Location Type field is set to zero, only a receiving device(s) that knows its absolute location SHALL respond by generating a Location Data Response command. If the Location Type field is set to one, all devices receiving this command SHALL respond by generating a Location Data Response command.

If the command is sent as a unicast, information for the device specified in the Target Address field SHALL be returned, if the receiving device has or can obtain the information for that device. If the information is not available, the Status field of the Location Data Response command SHALL be set to NOT_FOUND.

If the command is sent as a broadcast or multicast, receiving devices SHALL send back their own information (there is no IEEE target address in this case).

If the Number Responses field is greater than one, the subsequent location readings/calculations SHALL be sent using the Location Data Notification or the Compact Location Data Notification command, depending on the value of the Reduced Response field.

3.13.2.2.5 RSSI Response Command

This command is sent by a device in response to an RSSI Request command.

3.13.2.2.5.1 Payload Format

The command payload SHALL be formatted as illustrated in Figure 3-53.

Figure 3-53. Format of the RSSI Response Command Payload

Octets	8	2	2	2	1	1
Data Type	EUI64	int16	int16	int16	int8	uint8
Field Name	Replying Device	X	Y	Z	RSSI	*NumberRSSIMeasurements*

The fields of the payload have the following meanings:

Replying Device: The IEEE address of the neighbor that replies to the RSSI request

X, Y, Z: The coordinates of the replying node

RSSI: The RSSI registered by the replying node that refers to the radio link, expressed in dBm, between itself and the neighbor that performed the RSSI request

NumberRSSIMeasurements: How many packets were considered to give the RSSI value (=1 meaning no mean is supported)

3.13.2.2.5.2 Effect on Receipt

On receipt of this command, the server side of the location cluster will wait for *CalculationPeriod* time and generate a Report RSSI Measurement command.

3.13.2.2.6 Send Pings Command

This command is used to alert a node to start sending multiple packets so that all its one-hop neighbors can calculate the mean RSSI value of the radio link.

3.13.2.2.6.1 Payload Format

Send Pings command SHALL be formatted as illustrated in Figure 3-54. The address field contains the IEEE address of the node that have to perform the blasting (the destination node of this command) and the other fields of the payload correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions of the individual attributes.

Figure 3-54. Format of the Send Pings Command Payload

Octets	8	1	2
Data Type	EUI64	uint8	uint16
Field Name	Target Address	*NumberRSSIMeasurements*	*CalculationPeriod*

The Target Address field contains the IEEE address of the intended target node. This is included because there can be cases when the sender does not definitely know the short address of the intended target (see below for effect on receipt). The other fields of the payload correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions of the individual attributes.

3.13.2.2.6.2 Effect on Receipt

On receipt of this command, the device SHALL update the attributes corresponding to (i.e., with the same names as) the payload fields and generate a number of RSSI Ping commands equal to *NumberRSSIMeasurements* waiting for *CalculationPeriod* time between successive transmission of pings.

3.13.2.2.7 Anchor Node Announce Command

This command is sent by an anchor node when it joins the network, if it is already commissioned with the coordinates, to announce itself so that the central device knows the exact position of that device. This message SHOULD be either unicast to the central node or broadcast in the case that of unknown destination address.

3.13.2.2.7.1 Payload Format

Into the payload there are both the short and long addresses of the joining node as well as the coordinates of the node itself. 0xffff SHOULD be used if coordinates are not known.

The command payload SHALL be formatted as in Figure 3-55.

Figure 3-55. Format of the Anchor Node Announce Command Payload

Octets	8	2	2	2
Data Type	EUI64	int16	int16	int16
Field Name	Anchor Node IEEE Address	X	Y	Z

The Anchor Node Address field contains the IEEE address of the anchor node. The other fields of the payload correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions of the individual attributes. If any coordinate is unknown, it SHOULD be set to 0x8000.

3.13.2.3 Commands Generated

Table 3-77. Generated Command IDs for the RSSI Location Cluster

Command Identifier Field Value	Description	M/O
0x00	Device configuration response	M
0x01	Location data response	M
0x02	Location data notification	M
0x03	Compact location data notification	M
0x04	RSSI Ping	M
0x05	RSSI Request	O
0x06	Report RSSI Measurements	O

Command Identifier Field Value	Description	M/O
0x07	Request Own Location	O

3.13.2.3.1 Device Configuration Response Command

This command is sent by a device in response to a Get Device Configuration command (3.13.2.2.3).

3.13.2.3.1.1 Payload Format

The Device Configuration Response command payload SHALL be formatted as illustrated in Figure 3-56. All payload fields are relevant to the device for which the location parameters have been requested.

Figure 3-56. Format of the Device Configuration Response Payload

Octets	1	0 / 2	0 / 2	0 / 2	0 / 1	0 / 2
Data Type	enum8	int16	uint16	uint16	uint8	uint16
Field Name	Status	Power	Path Loss Exponent	Calculation Period	Number RSSI Measurements	Reporting Period

The fields of the payload (other than Status) correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions of the individual attributes.

The Status field indicates whether the response to the request was successful or not. If the field is set to SUCCESS, the response was successful. If the field is set to NOT_FOUND, the receiving device was unable to provide the location parameters of the device for which the location parameters were requested. If the field is set to NOT_FOUND, all other payload fields SHALL NOT be sent.

3.13.2.3.2 Location Data Response Command

This command is sent by a device in response to a request for location information and channel parameters.

3.13.2.3.2.1 Payload Format

The Location Data Response command payload SHALL be formatted as illustrated in Figure 3-57. All payload fields are relevant to the device for which the location parameters have been requested.

Figure 3-57. Format of the Location Data Response Payload

Octets	1	0 / 1	0 / 2	0 / 2	0 / 2	0 / 2	0 / 2	0 / 1	0 / 1	0 / 2
Data Type	enum8	data8	int16	int16	int16	int16	uint16	enum8	uint8	uint16
Field Name	Status	Location Type	Coordinate 1	Coordinate 2	Coordinate 3	Power	Path Loss Exponent	Location Method	Quality Measure	Location Age

The fields of the payload correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions of the individual attributes.

If the Absolute bit of the Location Type field is set to 1, the Location Method, Quality Measure and Location Age fields are not applicable and SHALL NOT be sent.

If the 2-D bit of the Location Type field is set to 1, the Coordinate 3 field SHALL NOT be sent.

The Status field indicates whether the response to the request was successful or not. If the field is set to SUCCESS, the response was successful. If the field is set to NOT_FOUND, the receiving device was unable to provide the location parameters of the device for which the location parameters were requested. If the field is set to NOT_FOUND, all other payload fields SHALL NOT be sent.

3.13.2.3.3 Location Data Notification Command

This command is sent periodically by a device to announce its location information and channel parameters. The period is equal to the value of the *ReportingPeriod* attribute.

The location data notification command MAY be sent as a unicast or as a broadcast frame. When sent as a broadcast frame, it is recommended that the broadcast radius is set to 1.

3.13.2.3.3.1 Payload Format

The Location Data Notification command payload SHALL be formatted as illustrated in Figure 3-58.

Figure 3-58. Format of the Location Data Notification Payload

Octets	1	2	2	0 / 2	2	2	0 / 1	0 / 1	0 / 2
Data Type	data8	int16	int16	int16	int16	uint16	enum8	uint8	uint16
Field Name	Location Type	Coordinate 1	Coordinate 2	Coordinate 3	Power	Path Loss Exponent	Location Method	Quality Measure	Location Age

The fields of the payload correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions of the individual attributes.

If the 2-D bit of the Location Type field is set to 1, the Coordinate 3 field SHALL NOT be sent.

If the Absolute bit of the Location Type field is set to 1, the Location Method, Quality Measure and Location Age fields are not applicable and SHALL NOT be sent.

3.13.2.3.4 Compact Location Data Notification Command

This command is identical in format and use to the Location Data Notification command, except that the Power, Path Loss Exponent and Location Method fields are not included.

3.13.2.3.5 RSSI Ping Command

This command is sent periodically by a device to enable listening devices to measure the received signal strength in the absence of other transmissions from that device. The period is given by the *ReportingPeriod* attribute.

The RSSI Ping command MAY be sent as a unicast or as a broadcast frame. When sent as a broadcast frame, it is recommended that the broadcast radius is set to 1.

3.13.2.3.5.1 Payload Format

The RSSI Ping command payload SHALL be formatted as illustrated in Figure 3-59.

Figure 3-59. Format of the RSSI Ping Command Payload

Octets	1
Data Type	data8
Field Name	Location Type

The Location Type field holds the value of the *LocationType* attribute.

3.13.2.3.6 RSSI Request Command

A device uses this command to ask one, more, or all its one-hop neighbors for the (mean) RSSI value they hear from itself.

3.13.2.3.6.1 Payload Format

The message is empty and MAY be used in broadcast (typical usage is broadcast with radius equal to one).

3.13.2.3.6.2 Effect on Receipt

On receipt of this command, the device SHALL respond by generating an RSSI Response command back to the sender of this request.

3.13.2.3.7 Report RSSI Measurements Command

This command is sent by a device to report its measurements of the link between itself and one or more neighbors. In a centralized location scenario, the device that sends this command is the device that needs to be localized.

3.13.2.3.7.1 Payload Format

The Report RSSI measurement command SHALL be formatted as in Figure 3-60.

Figure 3-60. Format of the Report RSSI Measurements Command Payload

Octets	8	1	N
Data Type	EUI64	uint8	Variable [E1]
Field Name	Measuring Device	N Neighbors	NeighborsInfo

NeighborsInfo structure is reported in Figure 3-61.

Figure 3-61. Neighbor Info Structure

Octets	8	2	2	2	1	1
Data Type	EUI64	int16	int16	int16	int8	uint8
Field Name	Neighbor	X	Y	Z	RSSI	NumberRSSI Measurements

The fields in the payload have the following meanings:

N Neighbors: Numbers of one-hop neighbours that reported the RSSI; indicates how many *NeighborsInfo* fields are present in the message.

Measuring Device: IEEE address of the device that report the measurements (i.e., the one that started the blast procedure)

Neighbors information:

X,Y,Z: Coordinates (if present) of the neighbor
Neighbor: IEEE address of the neighbor used to identify it if coordinates are either not present or not valid
RSSI: RSSI value registered by the neighbor that refer to the radio link between itself and measuring device
NumberRSSIMeasurements: How many packets were considered to give the RSSI value (=1 meaning is that no mean is supported)

3.13.2.3.8 Request Own Location Command

This command is sent by a node wishing to know its own location and it is sent to the device that performs the centralized localization algorithm.

3.13.2.3.8.1 Payload Format

The Request Own Location command payload SHALL be formatted as illustrated in Figure 3-62. The only field in the payload contains the IEEE address of the blind node, i.e., the node that wishes to know about its own location.

Figure 3-62. Format of the Request Own Location Command Payload

Octets	8
Data Type	EUI64
Field Name	IEEE Address of the Blind Node

3.13.2.3.8.2 Effect On Receipt

The node receiving the Request Own Location command will then reply with a Set Absolute Location command, telling the requesting entity its location.

3.13.3 Client

The client has no cluster specific attributes. The client generates the cluster-specific commands received by the server (see 3.13.2.2), as required by the application. The client receives the cluster-specific commands generated by the server (see 3.13.2.3).

3.14 Input, Output and Value Clusters

3.14.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This section specifies a number of clusters which are based on ‘Basic’ properties of the Input, Output and Value objects specified by BACnet (see [A1]).

The clusters specified herein are for use typically in Commercial Building applications, but MAY be used in any application domain.

3.14.2 Analog Input (Basic)

The Analog Input (Basic) cluster provides an interface for reading the value of an analog measurement and accessing various characteristics of that measurement. The cluster is typically used to implement a sensor that measures an analog physical quantity.

3.14.2.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

3.14.2.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AI	Type 2 (server to client)

3.14.2.3 Cluster Identifiers

Identifier	Name
0x000c	Analog Input

3.14.2.4 Server

3.14.2.4.1 Attributes

The attributes of this cluster are detailed in Table 3-78.

Table 3-78. Attributes of the Analog Input (Basic) Server Cluster

ID	Name	Type	Range	Acc	Default	M/O
0x001C	*Description*	string	-	R*W	Null string	O
0x0041	*MaxPresentValue*	single	-	R*W	-	O
0x0045	*MinPresentValue*	single	-	R*W	-	O
0x0051	*OutOfService*	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	*PresentValue*	single	-	RWP	-	M
0x0067	*Reliability*	enum8	-	R*W	0x00	O
0x006A	*Resolution*	single	-	R*W	-	O
0x006F	*StatusFlags*	map8	0x00 to 0x0f	RP	0	M
0x0075	*EngineeringUnits*	enum16	See section 3.14.11.10	R*W	-	O
0x0100	*ApplicationType*	uint32	0 to 0xffffffff	R	-	O

For an explanation of the attributes, see section 3.14.11.

3.14.2.5 Commands

No cluster specific commands are received or generated.

3.14.2.6 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands, according to the minimum and maximum reporting interval, value and timeout settings.

The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

3.14.2.7 Client

The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

3.14.3 Analog Output (Basic)

The Analog Output (Basic) cluster provides an interface for setting the value of an analog output (typically to the environment) and accessing various characteristics of that value.

3.14.3.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

3.14.3.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AO	Type 2 (server to client)

3.14.3.3 Cluster Identifiers

Identifier	Name
0x000d	Analog Output

3.14.3.4 Server

3.14.3.4.1 Attributes

The attributes of this cluster are detailed in Table 3-79.

Table 3-79. Attributes of the Analog Output (Basic) Server Cluster

Identifier	Name	Type	Range	Acces s	Default	M/O
0x001C	*Description*	string	-	R*W	Null string	O
0x0041	*MaxPresentValue*	single	-	R*W	-	O
0x0045	*MinPresentValue*	single	-	R*W	-	O

Identifier	Name	Type	Range	Acces s	Default	M/O
0x0051	*OutOfService*	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	*PresentValue*	single	-	RWP	-	M
0x0057	*PriorityArray*	Array of 16 structures of (bool, single)	-	RW	16 x (0, 0.0)	O
0x0067	*Reliability*	enum8	-	R*W	0x00	O
0x0068	*RelinquishDefault*	single	-	R*W	-	O
0x006A	*Resolution*	single	-	R*W	-	O
0x006F	*StatusFlags*	map8	0x00 to 0x0f	RP	0	M
0x0075	*EngineeringUnits*	enum16	See section 3.14.11.10	R*W	-	O
0x0100	*ApplicationType*	uint32	0 to 0xffffffff	R	-	O

For an explanation of the attributes, see section 3.14.11.

3.14.3.5 Commands

No cluster specific commands are received or generated.

3.14.3.6 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands, according to the minimum and maximum reporting interval, value and timeout settings.

The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

3.14.3.7 Client

The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

3.14.4 Analog Value (Basic)

The Analog Value (Basic) cluster provides an interface for setting an analog value, typically used as a control system parameter, and accessing various characteristics of that value.

3.14.4.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

3.14.4.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AV	Type 2 (server to client)

3.14.4.3 Cluster Identifiers

Identifier	Name
0x000e	Analog Value

3.14.4.4 Server

3.14.4.4.1 Attributes

The attributes of this cluster are detailed in Table 3-80.

Table 3-80. Attributes of the Analog Value (Basic) Server Cluster

Identifier	Name	Type	Range	Access	Default	M/O
0x001C	*Description*	string	-	R*W	Null string	O
0x0051	*OutOfService*	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	*PresentValue*	single	-	R/W	-	M
0x0057	*PriorityArray*	Array of 16 structures of (bool, single)	-	R/W	16 x (0, 0.0)	O
0x0067	*Reliability*	enum8	-	R*W	0x00	O
0x0068	*RelinquishDefault*	single	-	R*W	-	O
0x006F	*StatusFlags*	map8	0x00 to 0x0f	R	0	M
0x0075	*EngineeringUnits*	enum16	See section 3.14.11.10	R*W	-	O
0x0100	*ApplicationType*	uint32	0 to 0xffffffff	R	-	O

For an explanation of the attributes, see section 3.14.11.

3.14.4.5 Commands

No cluster specific commands are received or generated.

3.14.4.6 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands, according to the minimum and maximum reporting interval, value and timeout settings.

The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

3.14.4.7 Client

The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

3.14.5 Binary Input (Basic)

The Binary Input (Basic) cluster provides an interface for reading the value of a binary measurement and accessing various characteristics of that measurement. The cluster is typically used to implement a sensor that measures a two-state physical quantity.

3.14.5.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

3.14.5.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BI	Type 2 (server to client)

3.14.5.3 Cluster Identifiers

Identifier	Name
0x000f	Binary Input

3.14.5.4 Server

3.14.5.4.1 Attributes

The attributes of this cluster are detailed in Table 3-81.

Table 3-81. Attributes of the Binary Input (Basic) Server Cluster

Identifier	Name	Type	Range	Access	Default	M/O
0x0004	*ActiveText*	string	-	R*W	Null string	O
0x001C	*Description*	string	-	R*W	Null string	O
0x002E	*InactiveText*	string	-	R*W	Null string	O
0x0051	*OutOfService*	bool	False (0) or True (1)	R*W	False (0)	M
0x0054	*Polarity*	enum8	-	R	0	O
0x0055	*PresentValue*	bool	-	R*W	-	M
0x0067	*Reliability*	enum8	-	R*W	0x00	O
0x006F	*StatusFlags*	map8	0x00 to 0x0f	R	0	M
0x0100	*ApplicationType*	uint32	0 to 0xffffffff	R	-	O

For an explanation of the attributes, see section 3.14.11.

3.14.5.5 Commands

No cluster specific commands are received or generated.

3.14.5.6 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands, according to the minimum and maximum reporting interval, value and timeout settings.

The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

3.14.5.7 Client

The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

3.14.6 Binary Output (Basic)

The Binary Output (Basic) cluster provides an interface for setting the value of a binary output, and accessing various characteristics of that value.

3.14.6.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

3.14.6.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BO	Type 2 (server to client)

3.14.6.3 Cluster Identifiers

Identifier	Name
0x0010	Binary Output

3.14.6.4 Server

3.14.6.4.1 Attributes

The attributes of this cluster are detailed in Table 3-82.

Table 3-82. Attributes of the Binary Output (Basic) Server Cluster

ID	Name	Type	Range	Access	Default	MO
0x0004	*ActiveText*	string	-	R*W	Null string	O
0x001C	*Description*	string	-	R*W	Null string	O
0x002E	*InactiveText*	string	-	R*W	Null string	O
0x0042	*MinimumOffTime*	uint32	-	R*W	0xffffffff	O
0x0043	*MinimumOnTime*	uint32	-	R*W	0xffffffff	O
0x0051	*OutOfService*	bool	False (0) or True (1)	R*W	False (0)	M
0x0054	*Polarity*	enum8	-	R	0	O
0x0055	*PresentValue*	bool	-	R*W	-	M
0x0057	*PriorityArray*	Array of 16 structures of (bool, bool)	-	R/W	16 x (0, 0)	O
0x0067	*Reliability*	enum8	-	R*W	0x00	O
0x0068	*RelinquishDefault*	bool	-	R*W	-	O
0x006F	*StatusFlags*	map8	0x00 to 0x0f	R	0	M
0x0100	*ApplicationType*	uint32	0 to 0xffffffff	R	-	O

For an explanation of the attributes, see section 3.14.11.

3.14.6.5 Commands

No cluster specific commands are received or generated.

3.14.6.6 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands, according to the minimum and maximum reporting interval, value and timeout settings.

The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

3.14.6.7 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

3.14.7 Binary Value (Basic)

The Binary Value (Basic) cluster provides an interface for setting a binary value, typically used as a control system parameter, and accessing various characteristics of that value.

3.14.7.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

3.14.7.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BV	Type 2 (server to client)

3.14.7.3 Cluster Identifiers

Identifier	Name
0x0011	Binary Value

3.14.7.4 Server

3.14.7.4.1 Attributes

The attributes of this cluster are detailed in Table 3-83.

Table 3-83. Attributes of the Binary Value (Basic) Server Cluster

ID	Name	Type	Range	Access	Default	M/O
0x0004	*ActiveText*	string	-	R*W	Null string	O
0x001C	*Description*	string	-	R*W	Null string	O

ID	Name	Type	Range	Access	Default	M/O
0x002E	*InactiveText*	string	-	R*W	Null string	O
0x0042	*MinimumOffTime*	uint32	-	R*W	0xffffffff	O
0x0043	*MinimumOnTime*	uint32	-	R*W	0xffffffff	O
0x0051	*OutOfService*	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	*PresentValue*	bool	-	R*W	-	M
0x0057	*PriorityArray*	Array of 16 structures of (bool, bool)	-	R/W	16 x (0, 0)	O
0x0067	*Reliability*	enum8	-	R*W	0x00	O
0x0068	*RelinquishDefault*	bool	-	R*W	-	O
0x006F	*StatusFlags*	map8	0x00 to 0x0f	R	0	M
0x0100	*ApplicationType*	uint32	0 to 0xffffffff	R	-	O

For an explanation of the attributes, see section 3.14.11.

3.14.7.5 Commands

No cluster specific commands are received or generated.

3.14.7.6 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands, according to the minimum and maximum reporting interval, value and timeout settings.

The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

3.14.7.7 Client

The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

3.14.8 Multistate Input (Basic)

The Multistate Input (Basic) cluster provides an interface for reading the value of a multistate measurement and accessing various characteristics of that measurement. The cluster is typically used to implement a sensor that measures a physical quantity that can take on one of a number of discrete states.

3.14.8.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

3.14.8.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MI	Type 2 (server to client)

3.14.8.3 Cluster Identifiers

Identifier	Name
0x0012	Multistate Input

3.14.8.4 Server

3.14.8.4.1 Attributes

The attributes of this cluster are detailed in Table 3-84.

Table 3-84. Attributes of the Multistate Input (Basic) Server Cluster

Identifier	Name	Type	Range	Acc	Default	MO
0x000E	*StateText*	Array of character string	-	R*W	Null	O
0x001C	*Description*	string	-	R*W	Null string	O
0x004A	*NumberOfStates*	uint16	1 to 0xffff	R*W	0	M
0x0051	*OutOfService*	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	*PresentValue*	uint16	-	R*W	-	M
0x0067	*Reliability*	enum8	-	R*W	0x00	O
0x006F	*StatusFlags*	map8	0x00 to 0x0f	R	0	M
0x0100	*ApplicationType*	uint32	0 to 0xffffffff	R	-	O

For an explanation of the attributes, see section 3.14.11.

3.14.8.5 Commands

No cluster specific commands are received or generated.

3.14.8.6 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands, according to the minimum and maximum reporting interval, value and timeout settings.

The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

3.14.8.7 Client

The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

3.14.9 Multistate Output (Basic)

The Multistate Output (Basic) cluster provides an interface for setting the value of an output that can take one of a number of discrete values, and accessing characteristics of that value.

3.14.9.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

3.14.9.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MO	Type 2 (server to client)

3.14.9.3 Cluster Identifiers

Identifier	Name
0x0013	Multistate Output

3.14.9.4 Server

3.14.9.4.1 Attributes

The attributes of this cluster are detailed in Table 3-85.

Table 3-85. Attributes of the Multistate Output (Basic) Server Cluster

Identifier	Name	Type	Range	Access	Default	M/O
0x000E	*StateText*	Array of character string	-	R*W	Null	O
0x001C	*Description*	string	-	R*W	Null string	O
0x004A	*NumberOfStates*	uint16	1 to 0xffff	R*W	0	M
0x0051	*OutOfService*	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	*PresentValue*	uint16	-	R/W	-	M
0x0057	*PriorityArray*	Array of 16 structures of (bool, uint16)	-	R/W	16 x (0, 0)	O

Identifier	Name	Type	Range	Access	Default	M/O
0x0067	*Reliability*	enum8	-	R*W	0x00	O
0x0068	*RelinquishDefault*	uint16	-	R*W	-	O
0x006F	*StatusFlags*	map8	0x00 to 0x0f	R	0	M
0x0100	*ApplicationType*	uint32	0 to 0xffffffff	R	-	O

For an explanation of the attributes, see section 3.14.11.

3.14.9.5 Commands

No cluster specific commands are received or generated.

3.14.9.6 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands, according to the minimum and maximum reporting interval, value and timeout settings.

The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

3.14.9.7 Client

The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

3.14.10 Multistate Value (Basic)

The Multistate Value (Basic) cluster provides an interface for setting a multistate value, typically used as a control system parameter, and accessing characteristics of that value.

3.14.10.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

3.14.10.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MV	Type 2 (server to client)

3.14.10.3 Cluster Identifiers

Identifier	Name
0x0014	Multistate Value

3.14.10.4 Server

3.14.10.4.1 Attributes

The attributes of this cluster are detailed in Table 3-86.

Table 3-86. Attributes of the Multistate Value (Basic) Server Cluster

Identifier	Name	Type	Range	Access	Default	M/O
0x000E	*StateText*	Array of character string	-	R*W	Null	O
0x001C	*Description*	string	-	R*W	Null string	O
0x004A	*NumberOfStates*	uint16	1 to 0xffff	R*W	0	M
0x0051	*OutOfService*	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	*PresentValue*	uint16	-	R/W	-	M
0x0057	*PriorityArray*	Array of 16 structures of (bool, uint16)	-	R/W	16 x (0, 0)	O
0x0067	*Reliability*	enum8	-	R*W	0x00	O
0x0068	*RelinquishDefault*	uint16	-	R*W	-	O
0x006F	*StatusFlags*	map8	0x00 to 0x0f	R	0	M
0x0100	*ApplicationType*	uint32	0 to 0xffffffff	R	-	O

For an explanation of the attributes, see section 3.14.11.

3.14.10.5 Commands

No cluster specific commands are received or generated.

3.14.10.6 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands, according to the minimum and maximum reporting interval, value and timeout settings.

The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

3.14.10.7 Client

The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

3.14.11 Attribute Descriptions

Note: These attributes are based on BACnet properties with the same names. For more information, refer to the BACnet reference manual [A1].

3.14.11.1 OutOfService Attribute

The *OutOfService* attribute, of type Boolean, indicates whether (TRUE) or not (FALSE) the physical input, output or value that the cluster represents is not in service. For an Input cluster, when *OutOfService* is TRUE the *PresentValue* attribute is decoupled from the physical input and will not track changes to the physical input. For an Output cluster, when *OutOfService* is TRUE the *PresentValue* attribute is decoupled from the physical output, so changes to *PresentValue* will not affect the physical output. For a Value cluster, when *OutOfService* is TRUE the *PresentValue* attribute MAY be written to freely by software local to the device that the cluster resides on.

3.14.11.2 PresentValue Attribute

The *PresentValue* attribute indicates the current value of the input, output or value, as appropriate for the cluster. For Analog clusters it is of type single precision, for Binary clusters it is of type Boolean, and for multistate clusters it is of type Unsigned 16-bit integer.

The PresentValue attribute of an input cluster SHALL be writable when *OutOfService* is TRUE.

When the *PriorityArray* attribute is implemented, writing to *PresentValue* SHALL be equivalent to writing to element 16 of *PriorityArray*, i.e., with a priority of 16.

3.14.11.3 StatusFlags Attribute

This attribute, of type bitmap, represents four Boolean flags that indicate the general “health” of the analog sensor. Three of the flags are associated with the values of other optional attributes of this cluster. A more detailed status could be determined by reading the optional attributes (if supported) that are linked to these flags. The relationship between individual flags is not defined. The four flags are

Bit 0 = IN ALARM, Bit 1 = FAULT, Bit 2 = OVERRIDDEN, Bit 3 = OUT OF SERVICE

where:

IN ALARM - Logical FALSE (0) if the *EventState* attribute has a value of NORMAL, otherwise logical TRUE (1). This bit is always 0 unless the cluster implementing the *EventState* attribute is implemented on the same endpoint.

FAULT - Logical TRUE (1) if the *Reliability* attribute is present and does not have a value of NO FAULT DETECTED, otherwise logical FALSE (0).

OVERRIDDEN - Logical TRUE (1) if the cluster has been overridden by some mechanism local to the device. Otherwise, the value is logical FALSE (0).

In this context, for an input cluster, “overridden” is taken to mean that the *PresentValue* and *Reliability* (optional) attributes are no longer tracking changes to the physical input. For an Output cluster, “overridden” is taken to mean that the physical output is no longer tracking changes to the *PresentValue* attribute and the *Reliability* attribute is no longer a reflection of the physical output. For a Value cluster, “overridden” is taken to mean that the *PresentValue* attribute is not writeable.

OUT OF SERVICE - Logical TRUE (1) if the *OutOfService* attribute has a value of TRUE, otherwise logical FALSE (0).

3.14.11.4 Description Attribute

The *Description* attribute, of type Character string, MAY be used to hold a description of the usage of the input, output or value, as appropriate to the cluster. The character set used SHALL be ASCII, and the attribute SHALL contain a maximum of 16 characters, which SHALL be printable but are otherwise unrestricted.

3.14.11.5 MaxPresentValue Attribute

The *MaxPresentValue* attribute, of type Single precision, indicates the highest value that can be reliably obtained for the *PresentValue* attribute of an Analog Input cluster, or which can reliably be used for the *PresentValue* attribute of an Analog Output or Analog Value cluster.

3.14.11.6 PriorityArray Attribute

The *PriorityArray* attribute is an array of 16 structures. The first element of each structure is a Boolean, and the second element is of the same type as the *PresentValue* attribute of the corresponding cluster.

PriorityArray holds potential values for the *PresentValue* attribute of the corresponding cluster, in order of decreasing priority. The first value in the array corresponds to priority 1 (highest), the second value corresponds to priority 2, and so on, to the sixteenth value that corresponds to priority 16 (lowest).

The Boolean value in each element of the array indicates whether (TRUE) or not (FALSE) there is a valid value at that priority. All entries within the priority table are continuously monitored in order to locate the entry with the highest priority valid value, and *PresentValue* is set to this value.

When *PriorityArray* is supported, *PresentValue* MAY be written to indirectly by writing to the *PriorityArray*, as described above. If *PresentValue* is written to directly, a default priority of 16 (the lowest priority) SHALL be assumed, and the value is entered into the 16th element of *PriorityArray*.

When a value at a given priority is marked as invalid, by writing FALSE to its corresponding Boolean value, it is said to be relinquished.

(**Informative note:** In BACnet, each element of PriorityArray consists of a single value, which MAY be either of the same type as *PresentValue* or MAY be of type NULL to indicate that a value is not present. An attribute cannot have a variable data type; thus, an extra Boolean value is associated with each element of the array to indicate whether or not it is null).

3.14.11.7 RelinquishDefault Attribute

The *RelinquishDefault* attribute is the default value to be used for the *PresentValue* attribute when all elements of the *PriorityArray* attribute are marked as invalid.

3.14.11.8 MinPresentValue Attribute

The *MinPresentValue* attribute, of type Single precision, indicates the lowest value that can be reliably obtained for the *PresentValue* attribute of an Analog Input cluster, or which can reliably be used for the *PresentValue* attribute of an Analog Output or Analog Value cluster.

3.14.11.9 Reliability Attribute

The *Reliability* attribute, of type 8-bit enumeration, provides an indication of whether the *PresentValue* or the operation of the physical input, output or value in question (as appropriate for the cluster) is “reliable” as far as can be determined and, if not, why not. The *Reliability* attribute MAY have any of the following values:

NO-FAULT-DETECTED (0)

NO-SENSOR (1) - for input clusters only

OVER-RANGE (2)

UNDER-RANGE (3)

OPEN-LOOP (4)

SHORTED-LOOP (5)

NO-OUTPUT (6) - for input clusters only

UNRELIABLE-OTHER (7)

PROCESS-ERROR (8)

MULTI-STATE-FAULT (9) - for multistate clusters only

CONFIGURATION-ERROR (10)

3.14.11.10 EngineeringUnits Attribute

The *EngineeringUnits* attribute indicates the physical units associated with the value of the *PresentValue* attribute of an Analog cluster.

Values 0x0000 to 0x00fe are reserved for the list of engineering units with corresponding values specified in Clause 21 of the BACnet standard [A1]. 0x00ff represents 'other'. Values 0x0100 to 0xffff are available for proprietary use.

If the *ApplicationType* attribute is implemented, and is set to a value with a defined physical unit, the physical unit defined in *ApplicationType* takes priority over *EngineeringUnits*.

This attribute is defined to be Read Only, but a vendor can decide to allow this to be written to if *ApplicationType* is also supported. If this attribute is written to, how the device handles invalid units (e.g., changing Deg F to Cubic Feet per Minute), any local display or other vendor-specific operation (upon the change) is a local matter.

3.14.11.11 Resolution Attribute

This attribute, of type Single precision, indicates the smallest recognizable change to PresentValue.

3.14.11.12 ActiveText Attribute

This attribute, of type Character string, MAY be used to hold a human readable description of the ACTIVE state of a binary *PresentValue*. For example, for a Binary Input cluster, if the physical input is a switch contact, then the *ActiveText* attribute might be assigned a value such as “Fan 1 On”. If either the *ActiveText* attribute or the *InactiveText* attribute are present, then both of them SHALL be present.

The character set used SHALL be ASCII, and the attribute SHALL contain a maximum of 16 characters, which SHALL be printable but are otherwise unrestricted.

3.14.11.13 InactiveText Attribute

This attribute, of type Character string, MAY be used to hold a human readable description of the INACTIVE state of a binary *PresentValue*. For example, for a Binary Input cluster, if the physical input is a switch contact, then the *InactiveText* attribute might be assigned a value such as “Fan 1 Off”. If either the *InactiveText* attribute or the *ActiveText* attribute are present, then both of them SHALL be present.

The character set used SHALL be ASCII, and the attribute SHALL contain a maximum of 16 characters, which SHALL be printable but are otherwise unrestricted.

3.14.11.14 MinimumOffTime Attribute

This property, of type 32-bit unsigned integer, represents the minimum number of seconds that a binary *PresentValue* SHALL remain in the INACTIVE (0) state after a write to *PresentValue* causes it to assume the INACTIVE state.

3.14.11.15 MinimumOnTime Attribute

This property, of type 32-bit unsigned integer, represents the minimum number of seconds that a binary *PresentValue* SHALL remain in the ACTIVE (1) state after a write to *PresentValue* causes it to assume the ACTIVE state.

3.14.11.16 Polarity Attribute

This attribute, of type enumeration, indicates the relationship between the physical state of the input (or output as appropriate for the cluster) and the logical state represented by a binary *PresentValue* attribute, when *OutOfService* is FALSE. If the *Polarity* attribute is NORMAL (0), then the ACTIVE (1) state of the *PresentValue* attribute is also the ACTIVE or ON state of the physical input (or output). If the *Polarity* attribute is REVERSE (1), then the ACTIVE (1) state of the *PresentValue* attribute is the INACTIVE or OFF state of the physical input (or output).

Thus, when *OutOfService* is FALSE, for a constant physical input state a change in the *Polarity* attribute SHALL produce a change in the *PresentValue* attribute. If *OutOfService* is TRUE, then the *Polarity* attribute SHALL have no effect on the *PresentValue* attribute.

3.14.11.17 NumberOfStates Attribute

This attribute, of type Unsigned 16-bit integer, defines the number of states that a multistate *PresentValue* MAY have. The *NumberOfStates* property SHALL always have a value greater than zero. If the value of this property is changed, the size of the *StateText* array, if present, SHALL also be changed to the same value. The states are numbered consecutively, starting with 1.

3.14.11.18 StateText Attribute

This attribute, of type Array of Character strings, holds descriptions of all possible states of a multistate *PresentValue*. The number of descriptions matches the number of states defined in the *NumberOfStates* property. The *PresentValue*, interpreted as an integer, serves as an index into the array. If the size of this array is changed, the *NumberOfStates* property SHALL also be changed to the same value.

The character set used SHALL be ASCII, and the attribute SHALL contain a maximum of 16 characters, which SHALL be printable but are otherwise unrestricted.

3.14.11.19 ApplicationType Attribute

The *ApplicationType* attribute is an unsigned 32 bit integer that indicates the specific application usage for this cluster. (**Note:** This attribute has no BACnet equivalent.)

ApplicationType is subdivided into Group, Type and an Index number, as follows.

Group = Bits 24 to 31
An indication of the cluster this attribute is part of.

Type = Bits 16 to 23
For Analog clusters, the physical quantity that the Present Value attribute of the cluster represents. For Binary and Multistate clusters, the application usage domain.

Index = Bits 0 to 15
The specific application usage of the cluster.

3.14.11.19.1 Analog Input (AI) Types

Group = 0x00.

The following sub-clauses describe the values when Type = 0x00 to 0x0E. Types 0x0F to 0xFE are reserved, Type = 0xFF indicates other.

3.14.11.19.1.1 Type = 0x00: Temperature in degrees C

Table 3-87. AI Types, Type = 0x00: Temperature in Degrees C

Index	Application Usage
0x0000	2 Pipe Entering Water Temperature AI
0x0001	2 Pipe Leaving Water Temperature AI
0x0002	Boiler Entering Temperature AI
0x0003	Boiler Leaving Temperature AI
0x0004	Chiller Chilled Water Entering Temp AI
0x0005	Chiller Chilled Water Leaving Temp AI
0x0006	Chiller Condenser Water Entering Temp AI
0x0007	Chiller Condenser Water Leaving Temp AI
0x0008	Cold Deck Temperature AI
0x0009	Cooling Coil Discharge Temperature AI
0x000A	Cooling Entering Water Temperature AI
0x000B	Cooling Leaving Water Temperature AI
0x000C	Condenser Water Return Temperature AI
0x000D	Condenser Water Supply Temperature AI
0x000E	Decouple Loop Temperature AI
0x000F	Building Load AI
0x0010	Decouple Loop Temperature AI
0x0011	Dew Point Temperature AI
0x0012	Discharge Air Temperature AI
0x0013	Discharge Temperature AI
0x0014	Exhaust Air Temperature After Heat Recovery AI
0x0015	Exhaust Air Temperature AI
0x0016	Glycol Temperature AI
0x0017	Heat Recovery Air Temperature AI
0x0018	Hot Deck Temperature AI

Index	Application Usage
0x0019	Heat Exchanger Bypass Temp AI
0x001A	Heat Exchanger Entering Temp AI
0x001B	Heat Exchanger Leaving Temp AI
0x001C	Mechanical Room Temperature AI
0x001D	Mixed Air Temperature AI
0x001E	Mixed Air Temperature AI
0x001F	Outdoor Air Dewpoint Temp AI
0x0020	Outdoor Air Temperature AI
0x0021	Preheat Air Temperature AI
0x0022	Preheat Entering Water Temperature AI
0x0023	Preheat Leaving Water Temperature AI
0x0024	Primary Chilled Water Return Temp AI
0x0025	Primary Chilled Water Supply Temp AI
0x0026	Primary Hot Water Return Temp AI
0x0027	Primary Hot Water Supply Temp AI
0x0028	Reheat Coil Discharge Temperature AI
0x0029	Reheat Entering Water Temperature AI
0x002A	Reheat Leaving Water Temperature AI
0x002B	Return Air Temperature AI
0x002C	Secondary Chilled Water Return Temp AI
0x002D	Secondary Chilled Water Supply Temp AI
0x002E	Secondary HW Return Temp AI
0x002F	Secondary HW Supply Temp AI
0x0030	Sideloop Reset Temperature AI
0x0031	Sideloop Temperature Setpoint AI
0x0032	Sideloop Temperature AI
0x0033	Source Temperature

Index	Application Usage
0x0034	Supply Air Temperature AI
0x0035	Supply Low Limit Temperature AI
0x0036	Tower Basin Temp AI
0x0037	Two Pipe Leaving Water Temp AI
0x0038	Reserved
0x0039	Zone Dewpoint Temperature AI
0x003A	Zone Sensor Setpoint AI
0x003B	Zone Sensor Setpoint Offset AI
0x003C	Zone Temperature AI
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

5686 #### 3.14.11.19.1.2 Type = 0x01: Relative Humidity in %

5687 **Table 3-88. AI Types, Type = 0x01: Relative Humidity in %**

Index	Application Usage
0x0000	Discharge Humidity AI
0x0001	Exhaust Humidity AI
0x0002	Hot Deck Humidity AI
0x0003	Mixed Air Humidity AI
0x0004	Outdoor Air Humidity AI
0x0005	Return Humidity AI
0x0006	Sideloop Humidity AI
0x0007	Space Humidity AI
0x0008	Zone Humidity AI
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

5688 #### 3.14.11.19.1.3 Type = 0x02: Pressure in Pascal

5689 **Table 3-89. AI Types, Type = 0x02: Pressure in Pascal**

Index	Application Usage
0x0000	Boiler Pump Differential Pressure AI
0x0001	Building Static Pressure AI
0x0002	Cold Deck Differential Pressure Sensor AI
0x0003	Chilled Water Building Differential Pressure AI
0x0004	Cold Deck Differential Pressure AI
0x0005	Cold Deck Static Pressure AI
0x0006	Condenser Water Pump Differential Pressure AI
0x0007	Discharge Differential Pressure AI
0x0008	Discharge Static Pressure 1 AI
0x0009	Discharge Static Pressure 2 AI
0x000A	Exhaust Air Differential Pressure AI
0x000B	Exhaust Air Static Pressure AI
0x000C	Exhaust Differential Pressure AI
0x000D	Exhaust Differential Pressure AI
0x000E	Hot Deck Differential Pressure AI
0x000F	Hot Deck Differential Pressure AI
0x0010	Hot Deck Static Pressure AI
0x0011	Hot Water Bldg Diff Pressure AI
0x0012	Heat Exchanger Steam Pressure AI
0x0013	Minimum Outdoor Air Differential Pressure AI
0x0014	Outdoor Air Differential Pressure AI
0x0015	Primary Chilled Water Pump Differential Pressure AI
0x0016	Primary Hot water Pump Differential Pressure AI
0x0017	Relief Differential Pressure AI
0x0018	Return Air Static Pressure AI
0x0019	Return Differential Pressure AI

Index	Application Usage
0x001A	Secondary Chilled Water Pump Differential Pressure AI
0x001B	Secondary Hot water Pump Differential Pressure AI
0x001C	Sideloop Pressure AI
0x001D	Steam Pressure AI
0x001E	Supply Differential Pressure Sensor AI
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

5690 #### 3.14.11.19.1.4 Type = 0x03: Flow in Liters/Second

5691 **Table 3-90. AI Types, Type = 0x03: Flow in Liters/Second**

Index	Application Usage
0x0000	Chilled Water Flow AI
0x0001	Chiller Chilled Water Flow AI
0x0002	Chiller Condenser Water Flow AI
0x0003	Cold Deck Flow AI
0x0004	Decouple Loop Flow AI
0x0005	Discharge Flow AI
0x0006	Exhaust Fan Flow AI
0x0007	Exhaust Flow AI
0x0008	Fan Flow AI
0x0009	Hot Deck Flow AI
0x000A	Hot Water Flow AI
0x000B	Minimum Outdoor Air Fan Flow AI
0x000C	Minimum Outdoor Air Flow AI
0x000D	Outdoor Air Flow AI
0x000E	Primary Chilled Water Flow AI
0X000F	Relief Fan Flow AI

Index	Application Usage
0x0010	Relief Flow AI
0x0011	Return Fan Flow AI
0x0012	Return Flow AI
0x0013	Secondary Chilled Water Flow AI
0x0014	Supply Fan Flow AI
0x0015	Tower Fan Flow AI
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

5692 3.14.11.19.1.5 Type = 0x04: Percentage %

5693 **Table 3-91. AI Types, Type = 0x04: Percentage %**

Index	Application Usage
0x0000	Chiller % Full Load Amperage AI
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

5694 3.14.11.19.1.6 Type = 0x05: Parts per Million PPM

5695 **Table 3-92. AI types, Type = 0x05: Parts per Million PPM**

Index	Application Usage
0x0000	Return Carbon Dioxide AI
0x0001	Zone Carbon Dioxide AI
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

5696 3.14.11.19.1.7 Type = 0x06: Rotational Speed in RPM

Table 3-93. AI Types, Type = 0x06: Rotational Speed in RPM

Index	Application Usage
0x0000	Exhaust Fan Remote Speed AI
0x0001	Heat Recovery Wheel Remote Speed AI
0x0002	Min Outdoor Air Fan Remote Speed AI
0x0003	Relief Fan Remote Speed AI
0x0004	Return Fan Remote Speed AI
0x0005	Supply Fan Remote Speed AI
0x0006	Variable Speed Drive Motor Speed AI
0x0007	Variable Speed Drive Speed Setpoint AI
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.1.8 Type = 0x07: Current in Amps

Table 3-94. AI Types, Type = 0x07: Current in Amps

Index	Application Usage
0x0000	Chiller Amps AI
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.1.9 Type = 0x08: Frequency in Hz

Table 3-95. AI Types, Type = 0x08: Frequency in Hz

Index	Application Usage
0x0000	Variable Speed Drive Output Frequency AI
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.1.10 Type = 0x09: Power in Watts

Table 3-96. AI Types, Type = 0x09: Power in Watts

Index	Application Usage
0x0000	Power Consumption AI
0x0200- FFFE	Vendor defined
0xFFFF	Other

3.14.11.19.1.11 Type = 0x0A: Power in kW

Table 3-97. AI Types, Type = 0x0A: Power in kW

Index	Application Usage
0x0000	Absolute Power AI
0x0001	Power Consumption AI
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.1.12 Type = 0x0B: Energy in kWH

Table 3-98. AI Types, Type = 0x0B: Energy in kWH

Index	Application Usage
0x0000	Variable Speed Drive Kilowatt Hours AI
0x0200- FFFE	Vendor defined
0xFFFF	Other

3.14.11.19.1.13 Type = 0x0C: Count – Unitless

Table 3-99. AI Types, Type = 0x0C: Count - Unitless

Index	Application Usage
0x0000	Count
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.1.14 Type = 0x0D: Enthalpy in KJoules/Kg

Table 3-100. AI Types, Type = 0x0D: Enthalpy in KJoules/Kg

Index	Application Usage
0x0000	Outdoor Air Enthalpy AI
0x0001	Return Air Enthalpy AI
0x0002	Space Enthalpy
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.1.15 Type = 0x0E: Time in Seconds

Table 3-101. AI types, Type = 0x0E: Time in Seconds

Index	Application Usage
0x0000	Relative time AI
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2 Analog Output (AO) types

Group = 0x01.

The following sub-clauses describe the values when Type = 0x00 to 0x0E. Types 0x0F to 0xFE are reserved, Type = 0xFF indicates other.

3.14.11.19.2.1 Type = 0x00: Temperature in Degrees C

Table 3-102. AO Types, Type = 0x00: Temperature in Degrees C

Index	Application Usage
0x0000	Boiler AO
0x0001	Boiler Setpoint AO
0x0002	Cold Deck AO
0x0003	Chiller Setpoint AO
0x0004	Chiller Setpoint AO
0x0005	Hot Deck AO

Index	Application Usage
0x0006	Cooling Valve AO
0x0007	Zone Temperature Setpoint AO
0x0008	Setpoint Offset AO
0x0009	Setpoint Shift AO
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.2 Type = 0x01: Relative Humidity in %

Table 3-103. AO Types, Type = 0x01: Relative Humidity in %

Index	Application Usage
0x0000	Humidification AO
0x0001	Zone Relative Humidity Setpoint AO
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.3 Type = 0x02: Pressure Pascal

Table 3-104. AO Types, Type = 0x02: Pressure Pascal

Index	Application Usage
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.4 Type = 0x03: Flow in Liters/Second

Table 3-105. AO Types, Type = 0x03: Flow in Liters/Second

Index	Application Usage
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.5 Type = 0x04: Percentage %

5727 **Table 3-106. AO Types, Type = 0x04: Percentage %**

Index	Application Usage
0x0000	Face & Bypass Damper AO
0x0001	Heat Recovery Valve AO
0x0002	Heat Recovery Wheel AO
0x0003	Heating Valve AO
0x0004	Hot Deck Damper AO
0x0005	2 Pipe Damper AO
0x0006	2 Pipe Valve AO
0x0007	Boiler Mixing Valve AO
0x0008	Box Cooling Valve AO
0x0009	Box Heating Valve AO
0x000A	Chilled Water Bypass Valve AO
0x000B	Cold Deck Damper AO
0x000C	Cooling Damper AO
0x000D	Cooling Valve AO
0x000E	Damper AO
0x000F	Exhaust Air Damper AO
0x0010	Exhaust Damper AO
0x0011	Hot Water Bypass Valve AO
0x0012	Hot Water Mixing Valve AO
0x0013	Minimum Outside Air Damper AO
0x0014	Minimum Outside Air Fan AO
0x0015	Mixed Air Damper AO
0x0016	Mixing Valve AO
0x0017	Outside Air Damper AO

Index	Application Usage
0x0018	Primary Chilled Water Pump AO
0x0019	Primary Hot Water Pump AO
0x001A	Primary Heat Exchange Pump AO
0x001B	Preheat Damper AO
0x001C	Preheat Valve AO
0x001D	Reheat Valve 1 AO
0x001E	Reheat Valve AO
0x001F	Return Air Damper AO
0x0020	Secondary Chilled Water Pump AO
0x0021	Sequenced Valves AO
0x0022	Secondary Hot Water Pump AO
0x0023	Secondary Heat Exchange Pump AO
0x0024	Sideloop AO
0x0025	Supply Heating Valve AO
0x0026	Supply Damper AO
0x0027	Tower Bypass Valve AO
0x0028	Tower Fan AO
0x0029	Valve AO
0x002A	Zone 1 Damper AO
0x002B	Zone 1 Heating Valve AO
0x002C	Heat Recovery Exhaust Bypass Damper AO
0x002D	Heat Recovery Outside Air Bypass Damper AO
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

5728 ###### 3.14.11.19.2.6 Type = 0x05: Parts per Million PPM

Table 3-107. AO Types, Type = 0x05: Parts per Million PPM

Index	Application Usage
0x0000	Space Carbon Dioxide limit AO
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.7 Type = 0x06: Rotational Speed RPM

Table 3-108. AO Types, Type = 0x06: Rotational Speed RPM

Index	Application Usage
0x0000	Exhaust Fan Speed AO
0x0001	Fan Speed AO
0x0002	Relief Fan Speed AO
0x0003	Return Fan Speed AO
0x0004	Supply Fan Speed AO
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.8 Type = 0x07: Current in Amps

Table 3-109. AO Types, Type = 0x07: Current in Amps

Index	Application Usage
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.9 Type = 0x08: Frequency in Hz

Table 3-110. AO Types, Type = 0x08: Frequency in Hz

Index	Application Usage
0x0000 to 0x01FF	Reserved
0x0200- 0xFFFE	Vendor defined

Index	Application Usage
0xFFFF	Other

3.14.11.19.2.10 Type = 0x09: Power in Watts

Table 3-111. AO Types, Type = 0x09: Power in Watts

Index	Application Usage
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.11 Type = 0x0A: Power in kW

Table 3-112. AO Types, Type = 0x0A: Power in kW

Index	Application Usage
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.12 Type = 0x0B: Energy in kWh

Table 3-113. AO Types, Type = 0x0B: Energy in kWh

Index	Application Usage
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.13 Type = 0x0C: Count – Unitless

Table 3-114. AO Types, Type = 0x0C: Count - Unitless

Index	Application Usage
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.14 Type = 0x0D: Enthalpy in KJoules/Kg

Table 3-115. AO Types, Type = 0x0D: Enthalpy in KJoules/Kg

Index	Application Usage
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.2.15 Type = 0x0E: Time in Seconds

Table 3-116. AO Types, Type = 0x0E: Time in Seconds

Index	Application Usage
0x0000	Relative time AO
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.3 Analog Value (AV) Types

Group = 0x02.

The following sub-clauses describe the values when Type = 0x00 to 0x03. Types 0x04 to 0xFE are reserved, Type = 0xFF indicates other.

3.14.11.19.3.1 Type = 0x00: Temperature in Degrees C

Table 3-117. AV Types, Type = 0x00: Temperature in Degrees C

Index	Application Usage
0x0000	Setpoint Offset AV
0x0001	Temp Deadband AV
0x0002	Occupied Heating Setpoint AV
0x0003	Unoccupied Heating Setpoint AV
0x0004	Occupied Cooling Setpoint AV
0x0005	Unoccupied Cooling Setpoint AV
0x0006	Standby Heat Setpoint AV
0x0007	Standby Cooling Setpoint AV
0x0008	Effective Occupied Heating Setpoint AV

Index	Application Usage
0x0009	Effective Unoccupied Heating Setpoint AV
0x000a	Effective Occupied Cooling Setpoint AV
0x000b	Effective Unoccupied Cooling Setpoint AV
0x000c	Effective Standby Heat Setpoint AV
0x000d	Effective Standby Cooling Setpoint AV
0x000e	Setpoint Offset AV
0x000f	Setpoint Shift AV
0x0200 to fffe	Vendor defined
0xffff	Other

5754 #### 3.14.11.19.3.2 Type = 0x01: Area in Square Metres

5755 **Table 3-118. AV Types, Type = 0x01: Area in Square Metres**

Index	Application Usage
0x0000	Duct Area AV
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

5756 #### 3.14.11.19.3.3 Type = 0x02: Multiplier - Number

5757 **Table 3-119. AV Types, Type = 0x02: Multiplier - Number**

Index	Application Usage
0x0000	Gain multiplier AV
0x0200 to 0xfffe	Vendor defined
0xffff	Other

5758 #### 3.14.11.19.3.4 Type 0x03: Flow in Litres/Second

Table 3-120. AV Types, Type = 0x03: Flow in Litres/Second

Index	Application Usage
0x0000	Minimum Air Flow AV
0x0001	Maximum Air Flow AV
0x0002	Heating Minimum Air Flow AV
0x0003	Heating Maximum Air Flow AV
0x0004	Standby Minimum Air Flow AV
0x0005	Standby Maximum Air Flow AV
0x0200 to 0xfffe	Vendor defined
0xffff	Other

3.14.11.19.4 Binary Inputs (BI) Types

Group = 0x03.

The following sub-clauses describe the values when Type = 0x00 to 0x01. Types 0x02 to 0xFE are reserved, Type = 0xFF indicates other.

Present Value = 0 represents False, Off, Normal

Present Value = 1 represents True, On, Alarm

3.14.11.19.4.1 Type = 0x00: Application Domain HVAC

Table 3-121. BI Types, Type = 0x00: Application Domain HVAC

Index	Application Usage
0x0000	2 Pipe Pump Status BI
0x0001	Air Proving Switch BI
0x0002	Alarm Reset BI
0x0003	Boiler Status BI
0x0004	Boiler Flow Status BI
0x0005	Boiler General Alarm BI
0x0006	Boiler High Temperature Alarm BI
0x0007	Boiler Isolation Valve Status BI

Index	Application Usage
0x0008	Boiler Maintenance Switch BI
0x0009	Boiler Pump Overload BI
0x000A	Boiler Pump Status BI
0x000B	Boiler Status BI
0x000C	Box Heating Alarm BI
0x000D	Chiller Alarm BI
0x000E	Chiller Chilled Water Flow Status BI
0x000F	Chiller Chilled Water Isolation Valve Status BI
0x0010	Chiller Condenser Water Flow Status BI
0x0011	Chiller Condenser Water Isolation Valve Status BI
0x0012	Chiller Maintenance Switch BI
0x0013	Chiller Status BI
0x0014	Chilled Water Expansion Tank Alarm BI
0x0015	Chilled Water Expansion Tank High Pressure Alarm BI
0x0016	Chilled Water Expansion Tank Low Pressure Alarm BI
0x0017	Chilled Water Expansion Tank Status BI
0x0018	Combustion Damper Status BI
0x0019	Cooling Alarm BI
0x001A	Cooling Pump Maintenance Switch BI
0x001B	Cooling Pump Overload BI
0x001C	Cooling Pump Status BI
0x001D	Condenser Water Expansion Tank Alarm BI
0x001E	Condenser Water Expansion Tank High Pressure Alarm BI
0x001F	Condenser Water Expansion Tank Low Pressure Alarm BI
0x0020	Condenser Water Expansion Tank Status BI

Index	Application Usage
0x0021	Condenser Water Pump Maintenance Switch BI
0x0022	Condenser Water Pump Overload BI
0x0023	Condenser Water Pump Status BI
0x0024	Decouple Loop Flow Direction BI
0x0025	Discharge Smoke BI
0x0026	Door Status BI
0x0027	Economizer Command BI
0x0028	Emergency Shutdown BI
0x0029	Equipment Tamper BI
0x002A	Energy Hold Off BI
0x002B	Exhaust Fan Maintenance Switch BI
0x002C	Exhaust Fan Overload BI
0x002D	Exhaust Fan Status BI
0x002E	Exhaust Filter Status BI
0x002F	Exhaust Smoke BI
0x0030	Expansion Tank Alarm BI
0x0031	Expansion Tank High Pressure Alarm BI
0x0032	Expansion Tank Low Pressure Alarm BI
0x0033	Expansion Tank Status BI
0x0034	Fan Control By Others BI
0x0035	Fan Overload BI
0x0036	Filter Monitoring BI
0x0037	Final Filter Status BI
0x0038	Free Cooling Availability BI
0x0039	Heat Recovery Pump Status BI

Index	Application Usage
0x003A	Heat Recovery Wheel Alarm BI
0x003B	Heat Recovery Wheel Maintenance Switch BI
0x003C	Heat Recovery Wheel Overload BI
0x003D	Heat Recovery Wheel Status BI
0x003E	Heating Alarm BI
0x003F	Heating/Cooling Pump Maintenance Switch BI
0x0040	Heating/Cooling Pump Overload BI
0x0041	High Humidity Limit BI
0x0042	High Static Pressure Fault BI
0x0043	High Temperature Limit Fault BI
0x0044	Humidifier Alarm BI
0x0045	Humidifier Maintenance Switch BI
0x0046	Humidifier Overload BI
0x0047	Humidifier Status BI
0x0048	Heat Exchanger Alarm BI
0x0049	Heat Exchanger Isolation Valve Status BI
0x004A	Heat Exchanger Maintenance Switch BI
0x004B	Lighting Status BI
0x004C	Low Static Pressure Fault BI
0x004D	Low Temperature Limit Fault BI
0x004E	Minimum Outdoor Air Damper End Switch BI
0x004F	Minimum Outdoor Air Fan Maintenance Switch BI
0x0050	Minimum Outdoor Air Fan Overload BI
0x0051	Minimum Outdoor Air Fan Status BI
0x0052	Minimum Outdoor Air Fan Variable Frequency Drive Fault BI

Index	Application Usage
0x0053	Occupancy BI
0x0054	Occupancy Sensor BI
0x0055	Primary Chilled Water Pump Maintenance Switch BI
0x0056	Primary Chilled Water Pump Overload BI
0x0057	Primary Chilled Water Pump Status BI
0x0058	Primary Chilled Water Pump Maintenance Switch BI
0x0059	Primary Chilled Water Pump Overload BI
0x005A	Primary Chilled Water Pump Status BI
0x005B	Pre-Filter Status BI
0x005C	Preheat Alarm BI
0x005D	Preheat Bonnet Switch BI
0x005E	Preheat Pump Maintenance Switch BI
0x005F	Preheat Pump Overload BI
0x0060	Preheat Pump Status BI
0x0061	Refrigerant Alarm BI
0x0062	Reheat Alarm BI
0x0063	Reheat Bonnet Switch BI
0x0064	Reheat Pump Maintenance Switch BI
0x0065	Reheat Pump Overload BI
0x0066	Reheat Pump Status BI
0x0067	Relief Fan Maintenance Switch BI
0x0068	Relief Fan Overload BI
0x0069	Relief Fan Status BI
0x006A	Relief Fan Variable Frequency Drive Fault BI
0x006B	Return Air Smoke BI

Index	Application Usage
0x006C	Return Fan Maintenance Switch BI
0x006D	Return Fan Overload BI
0x006E	Return Fan Status BI
0x006F	Return Fan VFD Fault BI
0x0070	Return Smoke BI
0x0071	Secondary Chilled Water Pump 1 Maintenance Switch BI
0x0072	Secondary Chilled Water Pump 1 Overload BI
0x0073	Secondary Chilled Water Pump 1 Status BI
0x0074	Secondary Chilled Water Pump 1 Maintenance Switch BI
0x0075	Secondary Chilled Water Pump 1 Overload BI
0x0076	Secondary Chilled Water Pump 1 Status BI
0x0077	Sideloop BI
0x0078	Generic Status BI
0x0079	Summer Winter BI
0x007A	Supplemental Heating Alarm BI
0x007B	Supplemental Heating Pump Maintenance Switch BI
0x007C	Supplemental Heating Pump Overload BI
0x007D	Supplemental Heating Pump Status BI
0x007E	Supply Fan Maintenance Switch BI
0x007F	Supply Fan Overload BI
0x0080	Supply Fan Status BI
0x0081	Supply Fan Variable Frequency Drive Fault BI
0x0082	Temporary Occupancy BI
0x0083	Tower Level Alarm BI
0x0084	Tower Level Status BI

Index	Application Usage
0x0085	Tower Temp BI
0x0086	Tower Vibration Alarm Status BI
0x0087	Tower Level Alarm BI
0x0088	Tower Level Switch BI
0x0089	Tower Temp Switch BI
0x008A	Tower Fan Isolation Valve Status BI
0x008B	Tower Fan Maintenance Switch BI
0x008C	Tower Fan Overload BI
0x008D	Tower Fan Status BI
0x008E	Unit Enable BI
0x008F	Unit Reset BI
0x0090	Window Status BI
0x0091	Zone Sensor Temporary Occupancy BI
0x0092	Air Proving Switch BI
0x0093	Primary Heating Status BI
0x0094	Primary Cooling Status BI
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

5768 #### 3.14.11.19.4.2 Type = 0x01: Application Domain Security

5769 **Table 3-122. BI Types, Type = 0x01: Application Domain Security**

Index	Application Usage
0x0000	Glass Breakage Detection
0x0001	Intrusion Detection
0x0002	Motion Detection
0x0003	Glass Breakage Detection

Index	Application Usage
0x0004	Zone Armed
0x0005	Glass Breakage Detection
0x0006	Smoke Detection
0x0007	Carbon Dioxide Detection
0x0008	Heat Detection
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.5 Binary Output (BO) Types

Group = 0x04.

The following sub-clauses describe the values when Type = 0x00 to 0x01. Types 0x02 to 0xFE are reserved, Type = 0xFF indicates other.

Present Value = 0 represents False, Off, Normal

Present Value = 1 represents True, On, Alarm

3.14.11.19.5.1 Type = 0x00: Application Domain HVAC

Table 3-123. BO Types, Type = 0x00: Application Domain HVAC

Index	Application Usage
0x0000	2 Pipe Circulation Pump BO
0x0001	2 Pipe Valve BO
0x0002	2 Pipe Valve Command BO
0x0003	Boiler BO
0x0004	Boiler Isolation Valve BO
0x0005	Boiler Pump BO
0x0006	Box Cooling 2 Position BO
0x0007	Box Heating 2 Position BO
0x0008	Box Heating Enable BO
0x0009	Box Heating Stage 1 BO

Index	Application Usage
0x000A	Box Heating Stage 2 BO
0x000B	Box Heating Stage 3 BO
0x000C	Chiller 1 Isolation Valve BO
0x000D	Chiller BO
0x000E	Chiller Chilled Water Isolation Valve BO
0x000F	Chiller Condenser Water Isolation Valve BO
0x0010	Combustion Damper BO
0x0011	Compressor Stage 1 BO
0x0012	Compressor Stage 2 BO
0x0013	Cooling Circulation Pump BO
0x0014	Cooling Stage 1 BO
0x0015	Cooling Stage 2 BO
0x0016	Cooling Stage 3 BO
0x0017	Cooling Stage 4 BO
0x0018	Cooling Stage 5 BO
0x0019	Cooling Stage 6 BO
0x001A	Cooling Stage 7 BO
0x001B	Cooling Stage 8 BO
0x001C	Cooling Valve BO
0x001D	Cooling Valve Command BO
0x001E	Chilled Water Pump BO
0x001F	Economizer Enable BO
0x0020	Exhaust Air Damper BO
0x0021	Exhaust Fan BO
0x0022	Fan BO

Index	Application Usage
0x0023	Fan Speed 1 BO
0x0024	Fan Speed 2 BO
0x0025	Fan Speed 3 BO
0x0026	Heat Recovery Pump BO
0x0027	Heat Recovery Valve BO
0x0028	Heat Recovery Wheel BO
0x0029	Heating Stage 1 BO
0x002A	Heating Stage 2 BO
0x002B	Heating Stage 3 BO
0x002C	Heating Valve BO
0x002D	Heating Valve Command BO
0x002E	Hot Gas Bypass Valve BO
0x002F	Humidification Stage 1 BO
0x0030	Humidification Stage 2 BO
0x0031	Humidification Stage 3 BO
0x0032	Humidification Stage 4 BO
0x0033	Humidifier Enable BO
0x0034	Heat Exchanger Isolation Valve BO
0x0035	Lighting BO
0x0036	Minimum Outside Air Damper BO
0x0037	Minimum Outside Air Fan BO
0x0038	Outside Air Damper BO
0x0039	Primary Chilled Water Pump 1 BO
0x003A	Plate-and-Frame Heat Exchanger Isolation Valve BO
0x003B	Primary Hot Water Pump BO

Index	Application Usage
0x003C	Primary Heat Exchange Pump BO
0x003D	Preheat Circulation Pump BO
0x003E	Preheat Enable BO
0x003F	Preheat Stage 1 BO
0x0040	Preheat Stage 2 BO
0x0041	Preheat Stage 3 BO
0x0042	Preheat Stage 4 BO
0x0043	Preheat Stage 5 BO
0x0044	Preheat Stage 6 BO
0x0045	Preheat Stage 7 BO
0x0046	Preheat Stage 8 BO
0x0047	Preheat Valve BO
0x0048	Reheat Circulation Pump BO
0x0049	Reheat Enable BO
0x004A	Reheat Stage 1 BO
0x004B	Reheat Stage 2 BO
0x004C	Reheat Stage 3 BO
0x004D	Reheat Stage 4 BO
0x004E	Reheat Stage 5 BO
0x004F	Reheat Stage 6 BO
0x0050	Reheat Stage 7 BO
0x0051	Reheat Stage 8 BO
0x0052	Relief Fan BO
0x0053	Return Fan BO
0x0054	Reversing Valve 1 BO

Index	Application Usage
0x0055	Reversing Valve 2 BO
0x0056	Secondary Chilled Water Pump BO
0x0057	Secondary Hot Water Pump BO
0x0058	Secondary Heat Exchange Pump BO
0x0059	Sideloop BO
0x005A	Sideloop Stage 1 BO
0x005B	Sideloop Stage 2 BO
0x005C	Sideloop Stage 3 BO
0x005D	Sideloop Stage 4 BO
0x005E	Sideloop Stage 5 BO
0x005F	Sideloop Stage 6 BO
0x0060	Sideloop Stage 7 BO
0x0061	Sideloop Stage 8 BO
0x0062	Steam Isolation Valve BO
0x0063	Supplemental Heating 2 Position BO
0x0064	Supplemental Heating Stage 1 BO
0x0065	Supplemental Heating Valve BO
0x0066	Supplemental Heating Enable BO
0x0067	Supplemental Heating Pump BO
0x0068	Supply Fan BO
0x0069	Tower Basin Heater BO
0x006A	Tower Basin Makeup BO
0x006B	Tower Basin Heater BO
0x006C	Tower Basin Makeup BO
0x006D	Tower Isolation Valve BO

Index	Application Usage
0x006E	Tower Fan BO
0x006F	Tower Fan Speed 1 BO
0x0070	Tower Fan Speed 2 BO
0x0071	Tower Fan Speed 3 BO
0x0072	Zone Heating Stage 1 BO
0x0073	Zone Heating Stage 2 BO
0x0074	Zone Heating Stage 3 BO
0x0075	Zone Heating Valve BO
0x0076	2 Pipe Circulation Pump BO
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.5.2 Type = 0x02: Application Domain Security

Table 3-124. BO Types, Type = 0x02: Application Domain Security

Index	Application Usage
0x0000	Arm Disarm Command BO
0x0001	Occupancy Control BO
0x0002	Enable Control BO
0x0003	Access Control BO
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.6 Binary Value (BV) Types

Group = 0x05.

The following sub-clauses describe the values when Type = 0x00. Types 0x01 to 0xFE are reserved, Type = 0xFF indicates other.

Present Value = 0 represents False, Off, Normal

Present Value = 1 represents True, On, Alarm

3.14.11.19.6.1 Type = 0x00

Table 3-125. BV Types, Type = 0x00

Index	Application Usage
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.7 Multistate Input (MI) Types

Group = 0x0D.

The following sub-clauses describe the values when Type = 0x00. Types 0x01 to 0xFE are reserved, Type = 0xFF indicates other.

3.14.11.19.7.1 Type = 0x00: Application Domain HVAC

Table 3-126. MI Types, Type = 0x00: Application Domain HVAC

Index	Application Usage [Number of States] States
0x0000	[3] Off, On, Auto
0x0001	[4] Off, Low, Medium, High
0x0002	[7] Auto, Heat, Cool, Off, Emergency Heat, Fan Only, Max Heat
0x0003	[4] Occupied, Unoccupied, Standby, Bypass
0x0004	[3] Inactive, Active, Hold
0x0005	[8] Auto, Warm-up, Water Flush, Autocalibration, Shutdown Open, Shutdown Closed, Low Limit, Test and Balance
0x0006	[6] Off, Auto, Heat Cool, Heat Only, Cool Only, Fan Only
0x0007	[3] High, Normal, Low
0x0008	[4] Occupied, Unoccupied, Startup, Shutdown
0x0009	[3] Night, Day, Hold
0x000A	[5] Off, Cool, Heat, Auto, Emergency Heat
0x000B	[7] Shutdown Closed, Shutdown Open, Satisfied, Mixing, Cooling, Heating, Supplemental Heat

Index	Application Usage [Number of States] States
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

3.14.11.19.8 Multistate Output (MO) Types

Group = 0x0E.

The following sub-clauses describe the values when Type = 0x00. Types 0x01 to 0xFE are reserved, Type = 0xFF indicates other.

3.14.11.19.8.1 Type = 0x00: Application Domain HVAC

Table 3-127. MO Types, Type = 0x00: Application Domain HVAC

Index	Application Usage [Number of States] States
0x0000	[3] Off, On, Auto
0x0001	[4] Off, Low, Medium, High
0x0002	[7] Auto, Heat, Cool, Off, Emerg Heat, Fan Only, Max Heat
0x0003	[4] Occupied, Unoccupied, Standby, Bypass
0x0004	[3] Inactive, Active, Hold
0x0005	[8] Auto, Warm-up, Water Flush, Autocalibration, Shutdown Open, Shutdown Closed, Low Limit, Test and Balance
0x0006	[6] Off, Auto, Heat Cool, Heat Only, Cool Only, Fan Only
0x0007	[3] High, Normal, Low
0x0008	[4] Occupied, Unoccupied, Startup, Shutdown
0x0009	[3] Night, Day, Hold
0x000A	[5] Off, Cool, Heat, Auto, Emergency Heat
0x000B	[7] Shutdown Closed, Shutdown Open, Satisfied, Mixing, Cooling, Heating, Suppl Heat
0x0200- 0xFFFE	Vendor defined

Index	Application Usage [Number of States] States
0xFFFF	Other

3.14.11.19.9 Multistate Value (MV) Types

Group = 0x13.

The following sub-clauses describe the values when Type = 0x00. Types 0x01 to 0xFE are reserved, Type = 0xFF indicates other.

3.14.11.19.9.1 Type = 0x00: Application Domain HVAC

Table 3-128. MV Types, Type = 0x00: Application Domain HVAC

Index	Application Usage [Number of States] States
0x0000	[3] Off, On, Auto
0x0001	[4] Off, Low, Medium, High
0x0002	[7] Auto, Heat, Cool, Off, Emerg Heat, Fan Only, Max Heat
0x0003	[4] Occupied, Unoccupied, Standby, Bypass
0x0004	[3] Inactive, Active, Hold
0x0005	[8] Auto, Warm-up, Water Flush, Autocalibration, Shutdown Open, Shutdown Closed, Low Limit, Test and Balance
0x0006	[6] Off, Auto, Heat Cool, Heat Only, Cool Only, Fan Only
0x0007	[3] High, Normal, Low
0x0008	[4] Occupied, Unoccupied, Startup, Shutdown
0x0009	[3] Night, Day, Hold
0x000A	[5] Off, Cool, Heat, Auto, Emergency Heat
0x000B	[7] Shutdown Closed, Shutdown Open, Satisfied, Mixing, Cooling, Heating, Suppl Heat
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

All other group values are currently reserved

3.15 Diagnostics

3.15.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The diagnostics cluster provides access to information regarding the operation of the stack over time. This information is useful to installers and other network administrators who wish to know how a particular device is functioning on the network.

The Diagnostics Cluster needs to understand the performance of the network over time in order to isolate network routing issues.

While it is not absolutely essential, it is recommended that server attributes be stored in persistent memory. This especially makes sense if for instance some stack behavior were causing a device to reset. Without storing the associated server attributes in persistent memory there would be no way to analyze what was causing the reset behavior.

3.15.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added
2	CCB 2309 2212 2333

3.15.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	DIAG

3.15.1.3 Cluster Identifiers

Identifier	Name
0x0b05	Diagnostics

3.15.2 Server

3.15.2.1 Attributes

The server attributes in the diagnostics cluster are broken up into several attribute sets listed in Table 3-129.

Table 3-129. Server Attribute Sets of the Diagnostics Cluster

Attribute Set Identifier	Description
0x0000	Hardware Information
0x0100	Stack/Network Information

3.15.2.1.1 Hardware Information Attribute Set

Table 3-130. Hardware Information Attribute Set

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*NumberOfResets*	uint16	0x0000 to 0xffff	R	0x00000000	O
0x0001	*PersistentMemoryWrites*	uint16	0x0000 to 0xffff	R	0x00000000	O

3.15.2.1.1.1 *NumberOfResets* Attribute

An attribute that is incremented each time the device resets. A reset is defined as any time the device restarts. This is not the same as a reset to factory defaults, which SHOULD clear this and all values.

3.15.2.1.1.2 *PersistentMemoryWrites* Attribute

This attribute keeps track of the number of writes to persistent memory. Each time that the device stores a token in persistent memory it will increment this value.

3.15.2.1.2 Stack / Network Information Attribute Set

Note that many of the counters in this attribute set (Table 3-131) will wrap quickly. They SHOULD be read frequently during periods of network interrogation in order to avoid missing points where the counters roll over.

Table 3-131. Stack / Network Information Attribute Set

Id	Name	Type	Range	Acc	Def	MO
0x0100	*MacRxBcast*	uint32	0x00000000 to 0xffffffff[38]	R	0	O
0x0101	*MacTxBcast*	uint32	0x00000000 to 0xffffffff	R	0	O
0x0102	*MacRxUcast*	uint32	0x00000000 to 0xffffffff	R	0	O
0x0103	*MacTxUcast*	uint32	0x00000000 to 0xffffffff	R	0	O
0x0104	*MacTxUcastRetry*	uint16	0x0000 to 0xffff	R	0	O
0x0105	*MacTxUcastFail*	uint16	0x0000 to 0xffff	R	0	O
0x0106	*APSRxBcast*	uint16	0x0000 to 0xffff	R	0	O
0x0107	*APSTxBcast*	uint16	0x0000 to 0xffff	R	0	O

[38] CCB 2309

0x0108	*APSRxUcast*	uint16	0x0000 to 0xffff	R	0	O
0x0109	*APSTxUcastSuccess*	uint16	0x0000 to 0xffff	R	0	O
0x010A	*APSTxUcastRetry*	uint16	0x0000 to 0xffff	R	0	O
0x010B	*APSTxUcastFail*	uint16	0x0000 to 0xffff	R	0	O
0x010C	*RouteDiscInitiated*	uint16	0x0000 to 0xffff	R	0	O
0x010D	*NeighborAdded*	uint16	0x0000 to 0xffff	R	0	O
0x010E	*NeighborRemoved*	uint16	0x0000 to 0xffff	R	0	O
0x010F	*NeighborStale*	uint16	0x0000 to 0xffff	R	0	O
0x0110	*JoinIndication*	uint16	0x0000 to 0xffff	R	0	O
0x0111	*ChildMoved*	uint16	0x0000 to 0xffff	R	0	O
0x0112	*NWKFCFailure*	uint16	0x0000 to 0xffff	R	0	O
0x0113	*APSFCFailure*	uint16	0x0000 to 0xffff	R	0	O
0x0114	*APSUnauthorizedKey*	uint16	0x0000 to 0xffff	R	0	O
0x0115	*NWKDecryptFailures*	uint16	0x0000 to 0xffff	R	0	O
0x0116	*APSDecryptFailures*	uint16	0x0000 to 0xffff	R	0	O
0x0117	*PacketBufferAllocateFailures*	uint16	0x0000 to 0xffff	R	0	O
0x0118	*RelayedUcast*	uint16	0x0000 to 0xffff	R	0	O
0x0119	*PhytoMACqueuelimitreached*	uint16	0x0000 to 0xffff	R	0	O
0x011A	*PacketValidatedropcount*	uint16	0x0000 to 0xffff	R	0	O
0x011B	*AverageMACRetryPerAPSMessageSent*	uint16	0x0000 to 0xffff	R	0	O
0x011C	*LastMessageLQI*	uint8	0x00 to 0xff	R	0	O
0x011D	*LastMessageRSSI*	int8[39]	-127 to 127	R	0	O

5840 3.15.2.1.2.1 *MacRxBcast* Attribute

5841 A counter that is incremented each time the MAC layer receives a broadcast.

5842 3.15.2.1.2.2 *MacTxBcast* Attribute

[39] CCB 2212

A counter that is incremented each time the MAC layer transmits a broadcast.

3.15.2.1.2.3 *MacRxUcast* Attribute

A counter that is incremented each time the MAC layer receives a unicast.

3.15.2.1.2.4 *MacTxUcast* Attribute

A counter that is incremented each time the MAC layer transmits a unicast.

3.15.2.1.2.5 *MacTxUcastRetry* Attribute

A counter that is incremented each time the MAC layer retries a unicast.

3.15.2.1.2.6 *MacTxUcastFail* Attribute

A counter that is incremented each time the MAC layer fails to send a unicast.

3.15.2.1.2.7 *APSRxBcast* Attribute

A counter that is incremented each time the APS layer receives a broadcast.

3.15.2.1.2.8 *APSTxBcast* Attribute

A counter that is incremented each time the APS layer transmits a broadcast.

3.15.2.1.2.9 *APSRxUcast* Attribute

A counter that is incremented each time the APS layer receives a unicast.

3.15.2.1.2.10 *APSTxUcastSuccess* Attribute

A counter that is incremented each time the APS layer successfully transmits a unicast.

3.15.2.1.2.11 *APSTxUcastRetry* Attribute

A counter that is incremented each time the APS layer retries the sending of a unicast.

3.15.2.1.2.12 *APSTxUcastFail* Attribute

A counter that is incremented each time the APS layer fails to send a unicast.

3.15.2.1.2.13 *RouteDiscInitiated* Attribute

A counter that is incremented each time a route request is initiated[40].

3.15.2.1.2.14 *NeighborAdded* Attribute

A counter that is incremented each time an entry is added to the neighbor table.

3.15.2.1.2.15 *NeighborRemoved* Attribute

A counter that is incremented each time an entry is removed from the neighbor table.

3.15.2.1.2.16 *NeighborStale* Attribute

A counter that is incremented each time a neighbor table entry becomes stale because the neighbor has not been heard from.

3.15.2.1.2.17 *JoinIndication* Attribute

[40] CCB 2333

A counter that is incremented each time a node joins or rejoins the network via this node.

3.15.2.1.2.18 *ChildMoved* Attribute

A counter that is incremented each time an entry is removed from the child table.

3.15.2.1.2.19 *NWKFCFailure* Attribute

A counter that is incremented each time a message is dropped at the network layer because the APS frame counter was not higher than the last message seen from that source.

3.15.2.1.2.20 *APSFCFailure* Attribute

A counter that is incremented each time a message is dropped at the APS layer because the APS frame counter was not higher than the last message seen from that source.

3.15.2.1.2.21 *APSUnauthorizedKey* Attribute

A counter that is incremented each time a message is dropped at the APS layer because it had APS encryption but the key associated with the sender has not been authenticated, and thus the key is not authorized for use in APS data messages.

3.15.2.1.2.22 *NWKDecryptFailures* Attribute

A counter that is incremented each time a NWK encrypted message was received but dropped because decryption failed.

3.15.2.1.2.23 *APSDecryptFailures* Attribute

A counter that is incremented each time an APS encrypted message was received but dropped because decryption failed.

3.15.2.1.2.24 *PacketBufferAllocateFailures* Attribute

A counter that is incremented each time the stack failed to allocate a packet buffers. This doesn't necessarily mean that the packet buffer count was 0 at the time, but that the number requested was greater than the number free.

3.15.2.1.2.25 *RelayedUcast* Attribute

A counter that is incremented each time a unicast packet is relayed.

3.15.2.1.2.26 *PacketValidateDropCount* Attribute

A counter that is incremented each time a packet was dropped due to a packet validation error. This could be due to length or other formatting problems in the packet.

3.15.2.1.2.27 *AverageMACRetryPerAPSMessageSent* Attribute

A counter that is equal to the average number of MAC retries needed to send an APS message.

3.15.2.1.2.28 *LastMessageLQI* Attribute

This is the Link Quality Indicator for the last message received. There is no current agreed upon standard for calculating the LQI. For some implementations LQI is related directly to RSSI for others it is a function of the number of errors received over a fixed number of bytes in a given message. The one thing that has been agreed is that the Link Quality Indicator is a value between 0 and 255 where 0 indicates the worst possible link and 255 indicates the best possible link. Note that for a device reading the Last Message LQI the returned value SHALL be the LQI for the read attribute message used to read the attribute itself.

3.15.2.1.2.29 *LastMessageRSSI* Attribute

This is the receive signal strength indication for the last message received. As with Last Message LQI, a device reading the Last Message RSSI, the returned value SHALL be the RSSI of the read attribute message used to read the attribute itself.

3.15.2.2 Commands

There are no commands received by the server side of the diagnostics cluster.

3.15.3 Client

The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

3.16 Poll Control

3.16.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides a mechanism for the management of an end device's MAC Data Request rate. For the purposes of this cluster, the term "poll" always refers to the sending of a MAC Data Request from the end device to the end device's parent.

This cluster can be used for instance by a configuration device to make an end device responsive for a certain period of time so that the device can be managed by the controller.

This cluster is composed of a client and server. The end device implements the server side of this cluster. The server side contains several attributes related to the MAC Data Request rate for the device. The client side implements commands used to manage the poll rate for the device.

The end device which implements the server side of this cluster sends a query to the client on a predetermined interval to see if the client would like to manage the poll period of the end device in question. When the client side of the cluster hears from the server it has the opportunity to respond with configuration data to either put the end device in a short poll mode or let the end device continue to function normally.

3.16.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added; CCB 1815 1822 1833
2	CCB 2319 2329

3.16.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	POLL

3.16.1.3 Cluster Identifiers

Identifier	Name
0x0020	Poll Control

3.16.2 Terminology

MAC Data Request Rate: The MAC Data Request rate or simply "poll rate" is the frequency with which an end device sends a MAC Data Request to its parent. A parent device is only required to store a single message for its child for 7.68 seconds. Therefore if an end device wants to retrieve messages from its parent, it must send a MAC Data Request every 7.68 seconds.

Generally, end devices have two different rates at which they send MAC Data Polls to their parents. A slower rate for when the device is not expecting data (Long Poll Interval) and a faster rate (Short Poll Interval) for when the device is expecting data.

End devices only know that they are expecting data when they have initiated some sort of transaction. This cluster provides a mechanism for forcing this state to make the end device responsive to asynchronous messaging.

Long Poll Interval: The amount of time between MAC Data Requests when the device is in its normal operating state and not expecting any messages.

Short Poll Interval: The amount of time between MAC Data Requests when the device is either expecting data or has been put into "Fast Poll Mode" by the controlling device.

Fast Poll Mode: When the device is polling frequently to retrieve data from its parent we say that the device is in "Fast Poll Mode". The entire purpose of this cluster is to provide a means of managing when an end device goes into and out of Fast Poll Mode so that it can be made responsive for a controlling device.

3.16.3 Commissioning Process

Poll Control Cluster Clients SHALL configure bindings on the device implementing the Poll Control Cluster Server so that they will receive the regular check-in command on the configured *Check-In Interval*. This can be done during the configuration period on the end device implementing the Poll Control Cluster Server during which it is in fast poll mode. The device that implements the Poll Control Cluster Server SHALL check its bindings on the configured check-in Interval. If it has any bindings related to any endpoint and the Poll Control Cluster, it will send a check-in command out on that binding.

3.16.4 Server

3.16.4.1 Attributes

The server side of this cluster contains certain attributes (Table 3-132) associated with the poll period. *CheckInIntervalMin*, *LongPollIntervalMin*, *FastPollTimeoutMaximum* attributes are optional; however, if they are not supported, you could end up with a lot of chatter on the network as clients and servers attempt to negotiate the poll period. It is therefore recommended that these attributes be supported.

Table 3-132. Server Attributes

Identifier	Name	Type	Range	Acc	Default	M/O
0x0000	*Check-inInterval*	uint32	0x0 to 0x6E0000	RW	0x3840 (1 hr.)	M
0x0001	*LongPoll Interval*	uint32	0x04 to 0x6E0000	R	0x14 (5 sec)	M
0x0002	*ShortPollInterval*	uint16	0x01 to 0xffff	R	0x02 (2 qs)	M

Identifier	Name	Type	Range	Acc	Default	M/O
0x0003	*FastPollTimeout*	uint16	0x01 to 0xffff	RW	0x28 (10 sec.)	M
0x0004	*Check-inIntervalMin*	uint32	-	R	0	O
0x0005	*LongPollIntervalMin*	uint32	-	R	0	O
0x0006	*FastPollTimeoutMax*	uint16	-	R	0	O

3.16.4.1.1 *Check-inInterval* Attribute

The Poll Control server is responsible for checking in with the poll control client periodically to see if the poll control client wants to modify the poll rate of the poll control server. This is due to the fact that the Poll Control server is implemented on an end device that MAY have an unpredictable sleep-wake cycle.

The *Check-inInterval* represents the default amount of time between check-ins by the poll control server with the poll control client. The *Check-inInterval* is measured in quarterseconds. A value of 0 indicates that the Poll Control Server is turned off and the poll control server will not check-in with the poll control client.

The Poll Control Server checks in with the Poll Control Client by sending a Check-in command to the Client. This value SHOULD be longer than the *LongPoll Interval* attribute. If the Client writes an invalid attribute value (Example: Out of Range as defined in Table 3-132 or a value smaller than the optional *Check-inIntervalMin* attribute value or a value smaller than the *LongPollInterval* attribute value), the Server SHOULD return Write Attributes Response with an error status not equal to SUCCESS(0x00).

The Poll Control Client will hold onto the actions or messages for the Poll Control Server at the application level until the Poll Control Server checks in with the Poll Control Client.

3.16.4.1.2 *LongPollInterval* Attribute

An end device that implements the Poll Control server MAY optionally expose a *LongPollInterval* attribute. The Long Poll Interval represents the maximum amount of time in quarterseconds between MAC Data Requests from the end device to its parent.

The *LongPollInterval* defines the frequency of polling that an end device does when it is NOT in fast poll mode. The *LongPollInterval* SHOULD be longer than the *ShortPollInterval* attribute but shorter than the *Check-inInterval* attribute.

A value of 0xffffffff is reserved to indicate that the device does not have or does not know its long poll interval.

3.16.4.1.3 *ShortPollInterval* Attribute

An end device that implements the Poll Control server MAY optionally expose the *ShortPollInterval* attribute. The *ShortPollInterval* represents the number of quarterseconds that an end device waits between MAC Data Requests to its parent when it is expecting data (i.e., in fast poll mode).

3.16.4.1.4 FastPollTimeout Attribute

The *FastPollTimeout* attribute represents the number of quarterseconds that an end device will stay in fast poll mode by default. It is suggested that the *FastPollTimeout* attribute value be greater than 7.68 seconds.

The Poll Control Cluster Client MAY override this value by indicating a different value in the Fast Poll Duration argument in the Check-in Response command. If the Client writes a value out of range as defined in Table 3-132 or greater than the optional *FastPollTimeoutMax* attribute value if supported, the Server SHOULD return a Write Attributes Response with a status of INVALID_VALUE. An end device that implements the Poll Control server can be put into a fast poll mode during which it will send MAC Data Requests to its parent at the frequency of its configured *ShortPollInterval* attribute. During this period of time, fast polling is considered active. When the device goes into fast poll mode, it is required to send MAC Data Requests to its parent at an accelerated rate and is thus more responsive on the network and can receive data asynchronously from the device implementing the Poll Control Cluster Client.

3.16.4.1.5 *Check-inIntervalMin* Attribute

The Poll Control Server MAY optionally provide its own minimum value for the *Check-inInterval* to protect against the *Check-inInterval* being set too low and draining the battery on the end device implementing the Poll Control Server.

3.16.4.1.6 *LongPollIntervalMin* Attribute

The Poll Control Server MAY optionally provide its own minimum value for the *LongPollInterval* to protect against another device setting the value to too short a time resulting in an inadvertent power drain on the device.

3.16.4.1.7 *FastPollTimeoutMax* Attribute

The Poll Control Server MAY optionally provide its own maximum value for the *FastPollTimeout* to avoid it being set to too high a value resulting in an inadvertent power drain on the device.

3.16.4.2 Attribute Settings and Battery Life Considerations

The Poll Control Cluster is used on end devices that MAY be battery powered. In order to conserve battery life, it is important that the Poll Control Server maintain certain boundaries for the setting of the *Check-inInterval*, *LongPollInterval* and the *ShortPollInterval*. Therefore, while these attributes are all Readable and Writeable, it is possible that a battery-powered device might maintain its own boundary for the min and max of each of these attributes. The end device implementing the Poll Control Cluster Server MAY define its own boundaries for these attributes in order to protect itself against a power drain due to improper configuration.

For instance, a battery powered device MAY not allow another device to set its *Check-inInterval* to too short a value or its *FastPollTimeout* to too long an interval because it might cause the device to send too frequent check-in messages on the network and stay in fast poll mode for too long a time resulting in a drain on the battery.

The Check-inInterval, LongPollInterval and ShortPollInterval SHOULD be set such that:

Check-in Interval >= Long Poll Interval >= Short Poll Interval

The default values chosen for this cluster are:

Check-in Interval = 1 hour = 0x3840 quarterseconds

Long Poll Interval = 5 seconds = 0x14 quarterseconds

Short Poll Interval = 2 quarterseconds = 0x02 quarterseconds

Fast Poll Timeout = 10 seconds = 0x28 quarterseconds

Note that for the Check-in Interval, 0 is a special value and does not apply to this equation.

3.16.4.3 Commands

Table 3-133. Commands Generated by the Poll Control Server

Command ID	Description	Mandatory/Optional
0x00	Check-in	M

3.16.4.4 Check-in Command

The Poll Control Cluster server sends out a Check-in command to the devices to which it is paired based on the server's *Check-inInterval* attribute. It does this to find out if any of the Poll Control Cluster Clients with which it is paired are interested in having it enter fast poll mode so that it can be managed. This request is sent out based on either the *Check-inInterval*, or the next Check-in value in the Fast Poll Stop Request generated by the Poll Control Cluster Client.

The Check-in command expects a Check-in Response command to be sent back from the Poll Control Client. If the Poll Control Server does not receive a Check-in response back from the Poll Control Client up to7.68 seconds it is free to return to polling according to the *LongPollInterval*.

3.16.4.4.1 Payload Format

There is no payload for this command.

3.16.4.4.2 Effect on Receipt

Upon receipt of the Check-in command, the Poll Control Cluster client will respond with a Check-in Response command indicating that the server SHOULD or SHOULD not begin fast poll mode.

3.16.5 Client

3.16.5.1 Attributes

There are no attributes on the client side of the Poll Control Cluster.

3.16.5.2 Commands

Table 3-134. Commands Generated by the Poll Control Client

Command ID	Description	Mandatory/Optional
0x00	Check-in Response	M
0x01	Fast Poll Stop	M
0x02	Set Long Poll Interval	O
0x03	Set Short Poll Interval	O

3.16.5.3 Check-in Response Command

The Check-in Response is sent in response to the receipt of a Check-in command. The Check-in Response is used by the Poll Control Client to indicate whether it would like the device implementing the Poll Control Cluster Server to go into a fast poll mode and for how long. If the Poll Control Cluster Client indicates that it would like the device to go into a fast poll mode, it is responsible for telling the device to stop fast polling when it is done sending messages to the fast polling device.

If the Poll Control Server receives a Check-In Response from a client for which there is no binding (unbound), it SHOULD respond with a Default Response with a status value indicating ACTION_DENIED.

If the Poll Control Server receives a Check-In Response from a client for which there is a binding (bound) with an invalid fast poll timeout[41], it SHOULD respond with a Default Response with status INVALID_VALUE.

If the Poll Control Server receives a Check-In Response from a bound client after temporary fast poll mode is completed it SHOULD respond with a Default Response with a status value indicating TIMEOUT.

In all of the above cases, the Server SHALL respond with a Default Response not equal to SUCCESS[42].

3.16.5.3.1 Payload Format

Figure 3-63. Format of the Check-in Response Payload

Octets	1	2
Data Type	bool	uint16
Field Name	Start Fast Polling	Fast Poll Timeout

3.16.5.3.1.1 Start Fast Polling

This Boolean value indicates whether or not the Poll Control Server device SHOULD begin fast polling or not. If the Start Fast Polling value is true, the server device is EXPECTED to begin fast polling until the Fast Poll Timeout has expired. If the Start Fast Polling argument is false, the Poll Control Server MAY continue in normal operation and is not required to go into fast poll mode.

3.16.5.3.1.2 Fast Poll Timeout

The Fast Poll Timeout value indicates the number of quarterseconds during which the device SHOULD continue fast polling. If the Fast Poll Timeout value is 0, the device is EXPECTED to continue fast polling until the amount of time indicated it the *FastPollTimeout* attribute has elapsed or it receives a Fast Poll Stop command. If the Start Fast Polling argument is false, the Poll Control Server MAY ignore the Fast Poll Timeout argument.

The Fast Poll Timeout argument temporarily overrides the *FastPollTimeout* attribute on the Poll Control Cluster Server for the fast poll mode induced by the Check-in Response command. This value is not EXPECTED to overwrite the stored value in the *FastPollTimeout* attribute.

If the FastPollTimeout parameter in the CheckInResponse command is greater than the *FastPollTimeoutMax* attribute value, the Server Device SHALL respond with a default response of error status not equal to SUCCESS. It is suggested to use the Error Status of ZCL_INVALID_FIELD (0x85).

[41] CCB 2330
[42] CCB 2319 no such status code as ZCL_SUCCESS

3.16.5.4 Fast Poll Stop Command

The Fast Poll Stop command is used to stop the fast poll mode initiated by the Check-in response. The Fast Poll Stop command has no payload.

If the Poll Control Server receives a Fast Poll Stop from an unbound client it SHOULD send back a DefaultResponse with a value field indicating ACTION_DENIED" . The Server SHALL respond with a DefaultResponse not equal to SUCCESS.

If the Poll Control Server receives a Fast Poll Stop command from a bound client but it is unable to stop fast polling due to the fact that there is another bound client which has requested that polling continue it SHOULD respond with a Default Response with a status of "ACTION_DENIED"

If a Poll Control Server receives a Fast Poll Stop command from a bound client but it is not FastPolling it SHOULD respond with a Default Response with a status of ACTION_DENIED.

3.16.5.5 Set Long Poll Interval Command

The Set Long Poll Interval command is used to set the Read Only *LongPollInterval* attribute.

When the Poll Control Server receives the Set Long Poll Interval Command, it SHOULD check its internal minimal limit and the attributes relationship defined in 3.16.4.2 if the new Long Poll Interval is acceptable. If the new value is acceptable, the new value SHALL be saved to the *LongPollInterval* attribute. If the new value is not acceptable, the Poll Control Server SHALL send a default response of INVALID_VALUE (0x87) and the *LongPollInterval* attribute value is not updated.

3.16.5.5.1 Payload Format

Figure 3-64. Format of the Set Long Poll Interval Command Payload

Octets	4
Data Type	uint32
Field Name	NewLongPollInterval

3.16.5.6 Set Short Poll Interval Command

The Set Short Poll Interval command is used to set the Read Only *ShortPollInterval* attribute.

When the Poll Control Server receives the Set Short Poll Interval Command, it SHOULD check its internal minimal limit and the attributes relationship defined in 3.16.4.2 if the new Short Poll Interval is acceptable. If the new value is acceptable, the new value SHALL be saved to the *ShortPollInterval* attribute. If the new value is not acceptable, the Poll Control Server SHALL send a default response of INVALID_VALUE (0x87) and the *ShortPollInterval* attribute value is not updated.

3.16.5.6.1 Payload Format

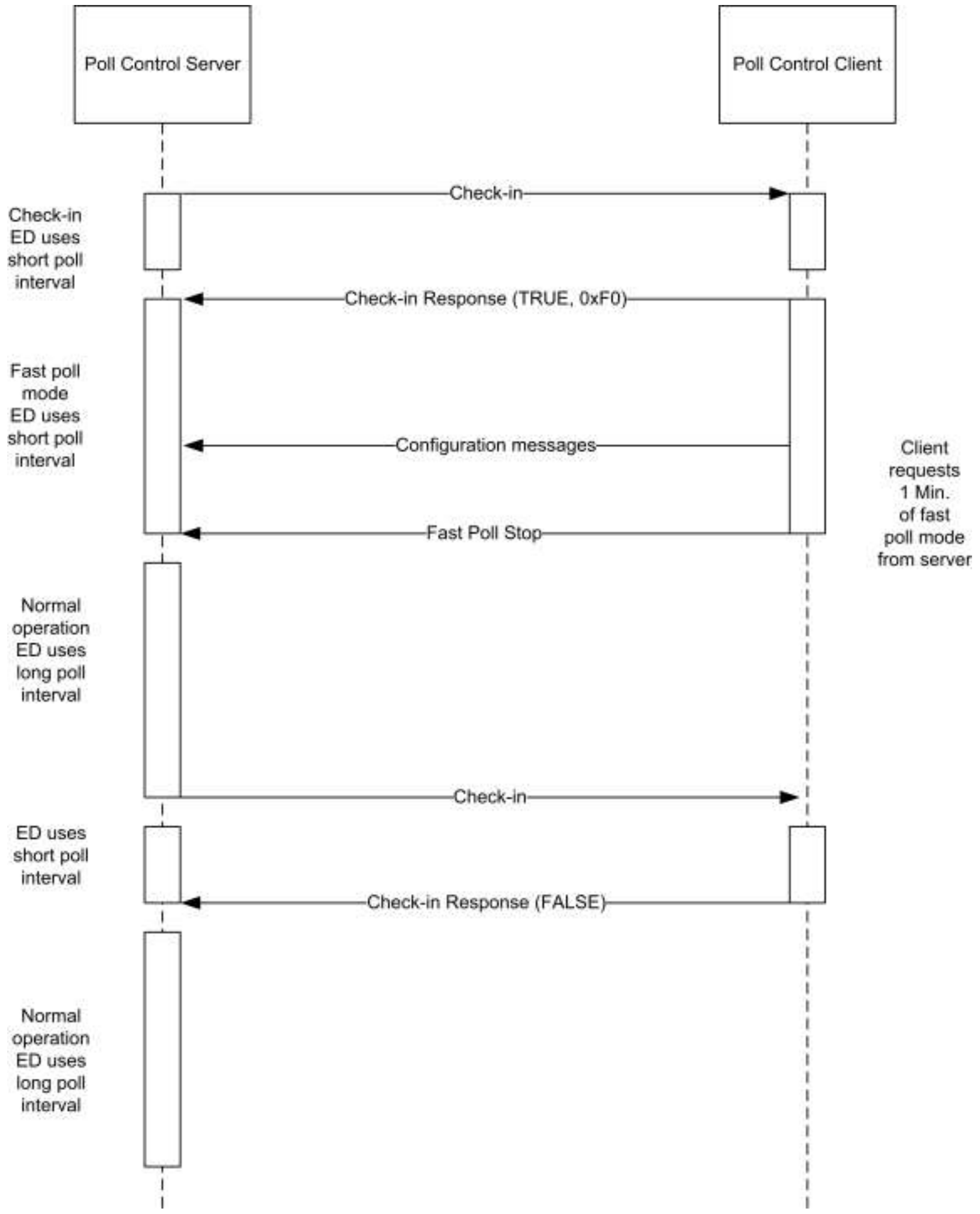
Figure 3-65. Format of the Set Short Poll Interval Command Payload

Octets	2
Data Type	uint16
Field Name	New Short Poll Interval

3.16.6 Poll Control Cluster Sequence Diagram

What follows is a typical sequence interaction between the client and server sides of the Poll Control Cluster.

Figure 3-66. Poll Control Cluster Sequence Diagram


Poll Control Server
Poll Control Client
Check-in
Check-in ED uses short poll interval
Check-in Response (TRUE, 0xF0)
Fast poll mode ED uses short poll interval
Configuration messages
Fast Poll Stop
Client requests 1 Min. of fast poll mode from server
Normal operation ED uses long poll interval
Check-in
ED uses short poll interval
Check-in Response (FALSE)
Normal operation ED uses long poll interval

3.16.6.1 Guaranteed Consistent Check-In Interval

Provided that the *Check-inInterval* attribute value stays constant, the interval between two Check In commands is guaranteed. The *Check-inInterval* SHOULD be kept independent regardless of when the Check-In Response or Fast Poll Stop command is received.

3.16.6.2 Multiple Poll Control Client

When the *Check-inInterval* expires, the Server SHOULD send parallel Check-In commands to all paired client devices.

The server SHOULD then enter a temporary Fast Poll Mode, with a fixed manufacturer-specific predefined check in timeout duration (t1), to wait for the Check-In Response Messages from all paired device.

Once the server received all the Check-In Response or if the temporary Fast Poll Mode timeout (t1), the server SHOULD then gather the information from all Check-In Response messages and determine the longest Fast Poll Timeout (t2) duration.

The Server device SHALL stay in the Fast Poll Mode for the longest Fast Poll Timeout (t2) duration. The server device MAY end fast poll mode before the longest fast poll timeout if it is able to determine that every start request from the paired device has been stopped explicitly by the Fast Poll Stop command or implicitly by a timeout.

For example:

Device A implements a poll control server, devices B and C implement poll control clients. Device A sends a check-in command to both B and C. Both B and C respond with check-in response command requesting a fast poll start. Assume B requests fast polling for 5 minutes and C requests fast polling for 10 minutes. If C sends a fast poll stop command after 7 minutes, device A MAY immediately end fast polling upon receipt of this command since the fast poll period requested by B would have expired after only 5 minutes (before the command from C was received).

3.16.6.3 Check-in Interval Attribute Changed

When the *Check-inInterval* attribute is changed (provided that the new value is valid and within acceptable range), the device SHOULD reset the internal check-in interval timer and send a check-in command according to the new *Check-inInterval* value.

3.17 Power Profile

This section describes the Power Profile cluster.

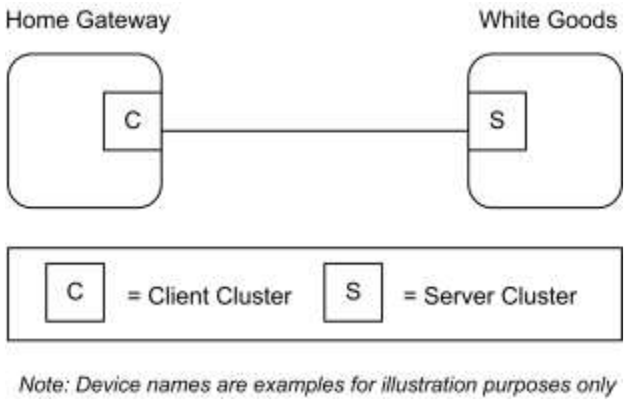
3.17.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface for transferring power profile information from a device (e.g., White Goods) to a controller (e.g., the Home Gateway). The Power Profile can be solicited by client side (request command) or can be notified directly from the device (server side). The Power Profile represents a forecast of the energy that a device is able to predict. It is split in multiple energy phases with a specific set of parameters representing the estimated “energy footprint” of an appliance. The data carried in the Power Profile can be updated during the different states of a Power Profile; since it represents a forecast of energy, duration and peak power of energy phases, it SHALL be considered as an estimation and not derived by measurements.

The Power Profile MAY also be used by an energy management system, together with other specific interfaces supported by the device, in order to schedule and control the device operation and to perform energy management within a home network. For more informative examples on how the Power Profile cluster might be used, see Chapter 15 Appliance Management, section 3.

Figure 3-67. Typical Usage of the Power Profile Cluster


Home Gateway
White Goods
C
S
C
= Client Cluster
S
= Server Cluster
Note: Device names are examples for illustration purposes only

3.17.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added

3.17.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	PWR

3.17.1.3 Cluster Identifiers

Identifier	Name
0x001a	Power Profile

3.17.2 References

The following standards and specifications contain provisions, which through reference in this document constitute provisions of this specification. All the standards and specifications listed are normative references. At the time of publication, the editions indicated were valid. All standards and specifications are subject to revision, and parties to agreements based on this specification are encouraged to investigate the possibility of applying the most recent editions of the standards and specifications indicated below.

3.17.3 General Description

3.17.3.1 Dependencies

The Power Profile Cluster is dependent upon the Appliance Control Cluster for the parts regarding the status notification and power management commands. Other specific clusters for actuation for devices different than Smart Appliances. Due to the possible length of the Power Profile commands, the devices supporting the Power Profile cluster MAY leverage on Partitioning if required by the application.

3.17.4 Server Attributes

. The following attributes represent the parameters for each Power Profile's phases.

Table 3-135. Attributes of the Power Profile Cluster

Id	Name	Type	Range	Access	Default	M/O
0x0000	*TotalProfileNum*	uint8	0x01 to 0xfe	R	1	M
0x0001	*MultipleScheduling*	bool	0x00 to 0x01	R	FALSE	M
0x0002	*EnergyFormatting*	map8	0x00 to 0xff	R	0x01*	M
0x0003	*EnergyRemote*	bool	TRUE or FALSE	R	FALSE	M
0x0004	*ScheduleMode*	map8	0x00 to 0xff	RWP	0x00	M

* 1/10 of Watt Hours represented

3.17.4.1 TotalProfileNum Attribute

The *TotalProfileNum* attribute represents the total number of profiles supported by the device. The minimum value for this attribute SHALL be 1.

3.17.4.2 MultipleScheduling Attribute

The *MultipleScheduling* attribute specifies if the server side of the Power Profile cluster supports the scheduling of multiple Energy Phases or it does support the scheduling of a single energy phase of the Power Profile at a time. If more than a single energy phases MAY be scheduled simultaneously the *MultipleScheduling* attribute SHALL be set to TRUE. In this case the device supporting the Power Profile server SHALL be able to process and manage scheduling commands carrying the schedule of more than one energy phase.

If the *MultipleScheduling* attribute is FALSE the device supporting the Power Profile client (e.g., EMS) SHALL be allowed to schedule just a single energy phase.

3.17.4.3 EnergyFormatting Attribute

The *EnergyFormatting* attribute provides a method to properly decipher the number of digits and the decimal location of the values found in the Energy Fields carried by the Power Profile Notification and Power Profile Response commands. This attribute is to be decoded as follows:

- Bits 0 to 2: Number of Digits to the right of the Decimal Point.
- Bits 3 to 6: Number of Digits to the left of the Decimal Point.
- Bit 7: If set, suppress leading zeros.

This attribute SHALL be used against the Energy fields.

3.17.4.4 EnergyRemote Attribute

The *EnergyRemote* attribute indicates whether the power profile server (e.g., appliance) is configured for remote control (e.g., by an energy management system).This refers to the selection chosen by the user on the remote control feature of the device. If the value is FALSE, the remote energy management is disabled, otherwise it is enabled. If the EnergyRemote is equal to FALSE all the supported PowerProfile SHALL set the Power Profile Remote Control field in the PowerProfile record equal to FALSE.

If the *EnergyRemote* attribute value is equal to TRUE, at least one PowerProfile SHALL be remotely controllable setting the Power Profile Remote Control field in the PowerProfile record to TRUE.

Table 3-136. *EnergyRemote* Attribute

Energy Remote Value	Description
0x00	FALSE = Remote Energy Management disabled
0x01	TRUE = Remote Energy Management enabled

3.17.4.5 ScheduleMode Attribute

The *ScheduleMode* attribute describes the criteria that SHOULD be used by the Power Profile cluster client side (e.g., energy management system) to schedule the power profiles.

3.17.4.5.1 Schedule Mode Field BitMap

Table 3-137. *ScheduleMode* Attribute

Bit	Description
bit0	bit0=1 : Schedule Mode Cheapest
bit1	bit1 =1: Schedule Mode Greenest
bit2 to bit7	Reserved

If the *ScheduleMode* attribute is set to the value 0x00, the scheduling criteria is demanded to the Power Profile cluster client side, which means that no specific preferences on the schedule mode are requested by the device supporting the server side of the power Profile cluster.

If “Schedule Mode Cheapest” is selected then the energy management system SHALL try to schedule the Power Profile to minimize the user’s energy bill.

If “Schedule Mode Greenest” is selected then the energy management system SHALL try to schedule the Power Profile to provide the highest availability of renewable energy sources.

Please note that how the energy management system MAY obtain “cheapest” or “greenest” information and estimate scheduling times is out of scope of this specification.

If more than a single bit is selected in the *ScheduleMode* bitmask, the Power Profile client SHALL try to calculate the schedule following all the selected criteria.

If all the bits are set to zero not specific optimization metrics preferences are requested by the device supporting the Power Profile server.

3.17.5 Server Commands Received

The command IDs for the commands received by the server side of the Power Profile Cluster are listed in Table 3-138.

Table 3-138. Cluster-Specific Commands Received by the Server

Command Identifier Field Value	Description	M/O
0x00	*PowerProfileRequest*	M

Command Identifier Field Value	Description	M/O
0x01	*PowerProfileStateRequest*	M
0x02	*GetPowerProfilePriceResponse*	M
0x03	*GetOverallSchedulePriceResponse*	M
0x04	*EnergyPhasesScheduleNotification*	M
0x05	*EnergyPhasesScheduleResponse*	M
0x06	*PowerProfileScheduleConstraintsRequest*	M
0x07	*EnergyPhasesScheduleStateRequest*	M
0x08	*GetPowerProfilePriceExtendedResponse*	M

3.17.5.1 PowerProfileRequest Command

The *PowerProfileRequest* Command is generated by a device supporting the client side of the Power Profile cluster in order to request the Power Profile of a server device. It is possible to request all profiles (without knowing how many Power Profiles the server has) or to request a specific PowerProfileID.

In the case of multiple profiles the server SHOULD send multiple messages, one for each Power Profile.

Although the profile is in a Power Profile running state (see *PowerProfileState*), the Power Profile Response transmitted as a reply to a *PowerProfileRequest* command SHALL carry all the energy phases of the estimated Power Profile, including the previous energy phases and the current energy phase which is running. The parameters of the Power Profile (e.g., the ExpectedDuration or the Energy fields of all the energy phases) MAY be updated for the same Power Profile due to a tuning in the forecast.

3.17.5.1.1 Payload Format

The *PowerProfileRequest* Command payload SHALL be formatted as illustrated in Figure 3-68.

Figure 3-68. Format of the *PowerProfileRequest* Command Payload

Octets	1
Data Type	uint8
Field Name	PowerProfileID

3.17.5.1.1.1 Payload Details

The payload of the *PowerProfileRequest* command carries the fields defined in Figure 3-68.

The PowerProfileID field specifies which profile (in the range 1 to *TotalProfileNum*) is requested. The special value 0x00 of this field does not refer to a particular profile; if 0x00 value is received the device SHOULD send details related to all the available profiles.

The PowerProfileID field SHALL NOT be greater than *TotalProfileNum*.

3.17.5.1.2 When Generated

This command is generated when the client side of the Power Profile cluster (e.g., a Home gateway device), needs to request the power profile to a device supporting the Power Profile cluster server side (e.g., White Good).

3.17.5.1.3 Effect on Receipt

The device that receives the Power Profile Request command SHALL reply with a *PowerProfileResponse* if supported. If the command is not supported the device SHALL reply with a standard ZCL Default response with status UNSUP_CLUSTER_COMMAND 0x81 (as from ZCL specification).

If the requested profile data are not available, the device SHALL reply with a standard ZCL response NOT_FOUND 0x8b (according to ZCL specification).

3.17.5.2 PowerProfileStateRequest Command

The *PowerProfileStateRequest* command is generated in order to retrieve the identifiers of current Power Profiles. This command does not have a payload.

3.17.5.2.1 Effect on Receipt

On receipt of this command, the device SHALL generate a *PowerProfileStateResponse* command.

3.17.5.3 GetPowerProfilePriceResponse Command

The *GetPowerProfilePriceResponse* command allows a device (client) to communicate the cost associated with a defined Power Profile to another device (server) requesting it. If the Price information requested related to the Power Profile is not available yet the response SHALL be a ZCL default response with "NOT FOUND" Status.

3.17.5.3.1 Payload Format

The *GetPowerProfilePriceResponse* command payload SHALL be formatted as illustrated in Figure 3-69.

Figure 3-69. Format of the *GetPowerProfilePriceResponse* Command

Octets	1	2	4	1
Data Type	uint8	uint16	uint32	uint8
Field Name	Power Profile ID	Currency	Price	Price Trailing Digit

3.17.5.3.1.1 Payload Details

PowerProfileID

The PowerProfileID field represents the identifier of the specific profile described by the Power Profile.

This is typically a sequential and contiguous number ranging from 1 to *TotalProfileNum*.

Currency

The Currency field identifies the local unit of currency used in the price field. This field is thought to be useful for displaying the appropriate symbol for a currency (i.e., $, €). The value of the currency field SHOULD match the values defined by ISO 4217.

Price

The Price field contains the price of the energy of a specific Power Profile measured in base unit of Currency per Unit of Measure (as described in the Metering Cluster, see SE specification) with the decimal point located as indicated by the PriceTrailingDigit field when the energy is delivered to the premise.

Price Trailing Digit

The PriceTrailingDigit field determines where the decimal point is located in the price field. The PriceTrailingDigit indicates the number of digits to the right of the decimal point.

3.17.5.3.2 When Generated

This command is generated when the command Get Power Profile Price is received. Please refer to Get Power Profile Price command description.

3.17.5.3.3 Effect on Receipt

On receipt of this command, the originator (server) is notified of the associated cost of the requested Power Profile, calculated by the client side of the Power Profile (see 9.7.10.1 for sequence diagrams and examples).

3.17.5.4 GetOverallSchedulePriceResponse Command

The *GetOverallSchedulePriceResponse* command allows a client to communicate the overall cost associated to all Power Profiles scheduled to a server requesting it. If the Price information requested is not available the response SHALL be a ZCL default response with “NOT FOUND” Status. The overall cost provided by the Power Profile Client side (e.g., energy management system) is intended as the cost of all the scheduled power profiles. This information MAY be helpful to assess the overall benefit provided by the scheduler, since a change in the scheduling of a specific device might -in some cases-increase its associated Power Profile cost. In fact in that case the schedule SHALL provide a global optimization by reducing the overall cost of all the scheduled power profiles, then reducing the energy bill for the user.

3.17.5.4.1 Payload Format

The Get Overall Schedule Price Response command payload SHALL be formatted as illustrated in Figure 3-70.

Figure 3-70. Format of the *GetOverallSchedulePriceResponse* Command

Octets	2	4	1
Data Type	uint16	uint32	uint8
Field Name	Currency	Price	Price Trailing Digit

3.17.5.4.2 Payload Details

See *GetPowerProfilePriceResponse* command payload details.

3.17.5.4.3 When Generated

This command is generated when the command *GetOverallSchedulePriceRequest* is received.

3.17.5.4.4 Effect on Receipt

On receipt of this command, the originator is notified of the overall cost of the scheduled Power Profiles, calculated by the Power Profile cluster client side. This information MAY be used to assess the overall benefit provided by the scheduler, which might be dependent on the schedule constraints. For more information, see Chapter 15 Appliance Management, section 3.

3.17.5.5 Energy Phases Schedule Notification Command

The Energy Phases Schedule Notification command is generated by a device supporting the client side of the Power Profile cluster in order to schedule the start of a Power Profile and its energy phases (they MAY be more than one in case of *MultipleScheduling* attribute equal to TRUE) on a the device supporting the server side of the Power Profile cluster, which did not solicit the schedule ("un-solicited" schedule). That happens when the Power Profile State carries a PowerProfileRemoteControl field equal to TRUE and the Energy Phase has a *MaxActivationDelay* different than 0x0000 (please note that changes on an already scheduled energy phase or power profile are possible but SHOULD be applied just in case of sensible advantages). The mechanisms designed to find the proper schedule are not part of the description of this command.

Please consider that, in case the *MultipleScheduling* attribute is FALSE (which means that the server side of the Power Profile cluster SHALL support the schedule of only a single energy phase at once), the Energy Phases Schedule Notification command SHOULD also be used to set a pause between two energy phases (energy pause behavior). In this case the Power Profile State MAY have any values but the command SHALL be issued only if the *PowerProfileRemoteControl* is set to TRUE and the Energy Phase has a *MaxActivationDelay* different than 0x0000.

3.17.5.5.1 Payload Format

The Energy Phases Schedule Notification command payload SHALL be formatted as illustrated in Figure 3-71.

Figure 3-71. Format of the *EnergyPhasesScheduleNotification* Command Payload

Octets	1	1	1	2	…	1	2
Data Type	uint8	uint8	uint8	uint16	…	uint8	uint16
Field Name	Power ProfileID	Num of Scheduled Phases	Energy PhaseID$_n$	Scheduled Time$_n$	…	Energy PhaseID$_n$	Scheduled Time$_n$

3.17.5.5.1.1 Payload Details

The payload of the *EnergyPhasesScheduleNotification* command carries the fields defined in Figure 3-71. Each *EnergyPhasesScheduleNotification* message SHALL include only one Power Profile and the energy phases of that Power Profile that needs to be scheduled. In case this command needs to be sent to a device supporting the server side of the power Profile Cluster with the *MultipleScheduling* attribute set to false, the payload of *EnergyPhasesScheduleNotification* command SHALL carry just one phase and the Scheduled Time field SHALL indicate the time scheduled for the whole Power Profile to start (in case the Power Profile is not started yet). If the Power Profile is in ENERGY_PHASE_RUNNING state and the server side of the cluster has the *MultipleScheduling* attribute set to false, the *EnergyPhasesScheduleNotification* command SHALL carry the scheduled time of the next energy phase.

PowerProfileID

See definition in *PowerProfileNotification* command.

Num of Scheduled Phases

The Num of Scheduled Phases field represents the total number of the energy phases of the Power Profile that need to be scheduled by this command.

The Energy phases that are not required to be scheduled SHALL NOT be counted in Num of Scheduled Phases field. The Num of Scheduled Phases SHALL be equal to 1 in case the *MultipleScheduling* attribute set to FALSE (only one energy phase SHALL be scheduled at a time). The *Num of Scheduled Phases* MAY be greater than 1in case the *MultipleScheduling* attribute set to TRUE (scheduling of multiple energy phases at the same time).

EnergyPhaseID

See definition in *PowerProfileNotification* command.

Scheduled Time

The Scheduled Time field represents the relative time scheduled in respect to the end of the previous energy phase. The unit is the minute. The Scheduled Time for the first Energy phase represents the scheduled time (expressed in relative encoding in respect to the current time) for the start of the Power Profile. The Scheduled Time fields for the subsequent Energy phases represent the relative time in minutes in respect to the previous scheduled Energy phase. The Energy phases that are not required to be scheduled will not be included in the commands and not be counted in Num of Scheduled Phases field. Only the Power Profile carrying a Power Profile Remote Control field equal to TRUE (as indicated in Power Profile State Notification command) and the Energy Phases supporting *MaxActivationDelay* different than 0x0000 SHALL be schedulable (as indicated in Power Profile Notification command).

3.17.5.5.2 When Generated

This command is generated when the client side of the Power Profile cluster (e.g., a Home gateway device), has calculated a specific schedule for a Power Profile and needs to send the schedule (i.e., “unsolicited” schedule) to a device supporting the Power Profile cluster server side (e.g., White Goods). This command SHALL be generated only if the recipient devices support schedulable Power Profiles (i.e., only if the Power Profile carries the first Energy Phase with a *MaxActivationDelay* different than 0x0000).

3.17.5.5.3 Effect on Receipt

The device that receives the *EnergyPhasesScheduleNotification* command SHALL reply with a standard Default response only if requested in the ZCL header of the *EnergyPhasesScheduleNotification* command or there is an error (as from ZCL specification).

If the device that receives the *EnergyPhasesScheduleNotification* command cannot schedule the energy phases because the activation delay of any of carried phases is equal to zero, it SHALL reply with a standard Default response with the error code NOT_AUTHORIZED (0x7e).

In case the scheduling state of the recipient entity changes after the reception of this command, the recipient will issue an Energy Phases Schedule State Notification.

3.17.5.6 EnergyPhasesScheduleResponse Command

This command is generated by the client side of Power Profile cluster as a reply to the *EnergyPhasesScheduleRequest* command.

3.17.5.6.1 Payload Format

The *EnergyPhasesScheduleResponse* command payload SHALL have the same payload as *EnergyPhasesScheduleNotification* command (*EnergyPhasesScheduleNotification* command, but “solicited” schedule because it is triggered by the *EnergyPhasesScheduleRequest* command). For more information, see Chapter 15, Appliance Management, section 3.

3.17.5.6.1.1 Payload Details

The payload of the *EnergyPhasesScheduleResponse* command carries the same fields as the *EnergyPhasesScheduleNotification* command. (*EnergyPhasesScheduleNotification* command, but “solicited” schedule because it is triggered by the *EnergyPhasesScheduleRequest* command)

3.17.5.6.2 When Generated

This command is generated when the server side of the Power Profile cluster (e.g., a White Goods device), has requested, using the *EnergyPhasesScheduleRequest*, the schedule of a specific power profile to a device supporting the Power Profile cluster client side (e.g., Home gateway) which SHALL calculate the schedules (“solicited” schedule) and reply with the *EnergyPhasesScheduleResponse*.

3.17.5.6.3 Effect on Receipt

The device that receives the *EnergyPhasesScheduleResponse* command SHALL reply with a standard Default response only if requested in the ZCL header of the *EnergyPhasesScheduleResponse* command. If the reception of *EnergyPhasesScheduleResponse* command is not supported the device SHALL reply with a standard ZCL Default response with status UNSUP_CLUSTER_COMMAND 0x81 (as from ZCL specification).

In case the scheduling state of the recipient entity changes after the reception of this command, the recipient will issue an *EnergyPhasesScheduleStateNotification*.

3.17.5.7 PowerProfileScheduleConstraintsRequest Command

The *PowerProfileScheduleConstraintsRequest* command is generated by client side of the Power Profile cluster in order to request the constraints of the Power Profile of a server, in order to set the proper boundaries for the scheduling when calculating the schedules.

3.17.5.7.1 Payload Format

The *PowerProfileScheduleConstraintsRequest* command payload is the same as the one used for *PowerProfileRequest* command. For more information, see Chapter 15, Appliance Management, section 3.

3.17.5.7.1.1 Payload Details

The payload of the *PowerProfileScheduleConstraintsRequest* command carries the fields defined in *PowerProfileRequest* command.

The Power Profile ID field specifies which profile (among *TotalProfileNum* total profiles number) the constraints are referring to.

3.17.5.7.2 When Generated

This command is generated when the client side of the Power Profile cluster (e.g., a Home gateway device), needs to request the constraints of the power profile to a device supporting the Power Profile cluster server side (e.g., Whitegood).

3.17.5.7.3 Effect on Receipt

The device that receives the Power Profile Schedule Constraints Request command SHALL reply with a Power Profile Schedule Constraints Response if supported. If the command is not supported, the device SHALL reply with a standard ZCL Default response with status UNSUP_CLUSTER_COMMAND 0x81 (as from ZCL specification).

If the requested profile data are not available, the device SHALL reply with a standard ZCL response NOT_FOUND 0x8b (according to ZCL specification).

3.17.5.8 EnergyPhasesScheduleStateRequest Command

The *EnergyPhasesScheduleStateRequest* command is generated by a device supporting the client side of the Power Profile cluster to check the states of the scheduling of a power profile, which is supported in the device implementing the server side of Power Profile cluster. This command can be used to re-align the schedules between server and client (e.g., after a client reset).

3.17.5.8.1 Payload Format

The *EnergyPhasesScheduleStateRequest* command payload is the same as the one used for *PowerProfileRequest* command. For more information, see Chapter 15, Appliance Management, section 3.

3.17.5.8.1.1 Payload Details

The payload of the *EnergyPhasesScheduleStateRequest* command carries the same fields defined in the *PowerProfileRequest* command. For more information, see Chapter 15, Appliance Management, section 3.

The Power Profile ID field specifies which profile (among *TotalProfileNum* total profiles number) the constraints are referring to.

3.17.5.8.2 When Generated

This command is generated when the client side of the Power Profile cluster (e.g., a Home gateway device), needs to check the schedules of the Power Profile to a device supporting the Power Profile cluster server side (e.g., White Good).

3.17.5.8.3 Effect on Receipt

The server that receives the *EnergyPhasesScheduleStateRequest* command SHALL reply to the client with an *EnergyPhasesScheduleStateResponse*, if supported. If the command is not supported, the servers SHALL reply with a standard ZCL Default response with status UNSUP_CLUSTER_COMMAND 0x81 (as from ZCL specification).

If the requested profile data are not available (e.g., invalid Power Profile ID), the server SHALL reply with a standard ZCL response NOT_FOUND 0x8b (according to ZCL specification).

If the server does not have any schedules set, it SHALL reply with a *EnergyPhasesScheduleStateResponse* carrying *NumofScheduledPhases* equal to zero (see Format of the *EnergyPhasesScheduleStateResponse* in case of no scheduled phases).

3.17.5.9 GetPowerProfilePriceExtendedResponse Command

The *GetPowerProfilePriceExtendedResponse* command allows a device (client) to communicate the cost associated to all Power Profiles scheduled to another device (server) requesting it according to the specific options contained in the *EnergyPhasesScheduleStateResponse*. If the Price information requested is not available, the response SHALL be a ZCL default response with "NOT FOUND" Status.

3.17.5.9.1 Payload Format

The *EnergyPhasesScheduleStateResponse* command payload SHALL be formatted as the *GetPowerProfilePriceResponse* command.

3.17.5.9.2 Payload Details

See *GetPowerProfilePriceResponse* command payload details.

3.17.5.9.3 When Generated

This command is generated when the command *GetPowerProfilePriceExtendedResponse* is received.

3.17.5.9.4 Effect on Receipt

On receipt of this command, the originator is notified of cost of the scheduled Power Profiles, calculated by the Power Profile cluster server side according to the specific option transmitted in the *EnergyPhasesScheduleStateResponse* command (e.g., cost at specific PowerProfileStartTime). For more information, see Chapter 15, Appliance Management, section 3.

3.17.6 Server Commands Generated

Cluster-specific commands are generated by the server, as shown in Table 3-139.

Table 3-139. Cluster-Specific Commands Sent by the Server

Command Identifier Field Value	Description	M/O
0x00	*PowerProfileNotification*	M
0x01	*PowerProfileResponse*	M
0x02	*PowerProfileStateResponse*	M
0x03	*GetPowerProfilePrice*	O
0x04	*PowerProfilesStateNotification*	M
0x05	*GetOverallSchedulePrice*	O
0x06	*EnergyPhasesScheduleRequest*	M
0x07	*EnergyPhasesScheduleStateResponse*	M
0x08	*EnergyPhasesScheduleStateNotification*	M
0x09	*PowerProfileScheduleConstraintsNotification*	M
0x0A	*PowerProfileScheduleConstraintsResponse*	M
0x0B	*GetPowerProfilePriceExtended*	O

3.17.6.1 PowerProfileNotification Command

The *PowerProfileNotification* command is generated by a device supporting the server side of the Power Profile cluster in order to send the information of the specific parameters (such as Peak power and others) belonging to each phase.

3.17.6.1.1 Payload Format

The *PowerProfileNotification* command payload SHALL be formatted as illustrated in Figure 3-72.

Figure 3-72. Format of the *PowerProfileNotification* Command Payload (1 of 2)

Octets	1	1	1	1	1	2	2	2
Data Type	uint8	uint8	uint8	uint8	uint8	uint16	uint16	uint16
Field Name	*Total Profile Num*	*Power ProfileID*	*Num of Transferred Phases*	*Energy PhaseID$_1$*	*Macro PhaseID$_1$*	*ExpectedDuration$_1$*	*Peak Power$_1$*	*Energy$_1$*

Octets	2	…	1	1	2	2	2	2
Data Type	uint16	…	uint8	uint8	uint16	uint16	uint16	uint16
Field Name	*MaxActivationDelay$_1$*	…	*Energy PhaseID$_n$*	*Macro PhaseID$_n$*	*Expected Duration$_n$*	*Peak Power$_n$*	*Energy$_n$*	*MaxActivationDelay$_n$*

3.17.6.1.1.1 Payload Details

The payload of the *PowerProfileNotification* command carries the fields defined in Figure 3-72. Each *PowerProfileNotification* message SHALL include only one Power Profile.

If multiple phases are transferred within a single *PowerProfileNotification* command (i.e., *Number of Transferred Phases* greater than 1), the parameters of the other phases (*PhaseID*, *ExpectedDuration*, etc.) SHOULD be carried in the payload. Each phase has a fixed number of parameters and the total length is 10 octets, so that the total length of the payload could be calculated with the following formula:

*Total Payload Length = 1 + 1 + 1 + (Num of Transferred Phases * 10)*

TotalProfileNum

For more information, see Chapter 15, Appliance Management, section 3.

PowerProfileID

The PowerProfileID field represents the identifier of the specific profile described by the Power Profile.

This field contains a sequential and contiguous number ranging from 1 to *TotalProfileNum.*

Num of Transferred Phases

This field represents the number of the energy phases of the Power Profile.

MacroPhaseID

The MacroPhaseID field represents the identifier of the specific phase (operational-displayed) described by the Power Profile.

This reference could be used in conjunction with a table of ASCII strings, describing the label of the functional phase. This table is not described in the contest of the Power Profile because it MAY be not functionally linked with energy management.

EnergyPhaseID

The EnergyPhaseID field indicates the identifier of the specific energy phase described by the Power Profile.

This is a sequential and contiguous number ranging from 1 to the maximum number of phases belonging to the Power Profile.

The value 0xFF SHALL be used to specify invalid energy phase (e.g., for a Power Profile in IDLE state).

ExpectedDuration

The ExpectedDuration field represents the estimated duration of the specific phase. Each unit is a minute.

PeakPower

The PeakPower field represents the estimated power for the specific phase. Each unit is a Watt.

Energy

The Energy field represents the estimated energy consumption for the accounted phase. Each unit is Watt per hours, according to the formatting specified in the *EnergyFormatting* attribute. For more information, see Chapter 15, Appliance Management, section 3. The Energy value fulfills the following equation:

$Energy \leq PeakPower(Watt) * ExpectedDuration(sec)$

MaxActivationDelay

The MaxActivationDelay field indicates the maximum interruption time between the end of the previous phase and the beginning of the specific phase. Each unit is a minute.

The special value 0x0000 means that it is not possible to insert a pause between the two consecutive phases.

The MaxActivationDelay field of the first energy phase of a Power Profile SHALL be set to the value 0xFFFF.

3.17.6.1.2 When Generated

This command is generated when the server side of the Power Profile cluster (e.g., a White Good device), need to send the representation of its power profile to a controller device supporting the Power Profile cluster client side (e.g., Home Gateway).

3.17.6.1.3 Effect on Receipt

The device that receives the *PowerProfileNotification* command SHALL reply with a standard Default response if requested in the ZCL header of the *PowerProfileNotification* command.

3.17.6.2 PowerProfileResponse Command

This command is generated by the server side of Power Profile cluster as a reply to the *PowerProfileRequest* command. If the reception of *PowerProfileRequest* command is not supported the device SHALL reply with a standard ZCL Default response with status UNSUP_CLUSTER_COMMAND 0x81 (as from ZCL specification).

If the profile data requested are not available, the device SHALL reply with a standard ZCL response INVALID_VALUE 0x87 (as ZCL specification).

3.17.6.2.1 Payload Format

The *PowerProfileResponse* Command payload SHALL be formatted as illustrated in Figure 3-72 (same as *PowerProfileNotification* command).

3.17.6.2.1.1 Payload Details

The payload of the *PowerProfileResponse* command carries the fields defined in Figure 3-72 (the same as *PowerProfileNotification* command).

3.17.6.2.2 When Generated

This command is generated by the server side of Power Profile cluster (e.g., White Good) as a reply to the *PowerProfileRequest* command sent by the client side (e.g., a Home gateway device).

3.17.6.2.3 Effect on Receipt

The device that receives the *PowerProfileResponse* command SHALL reply with a standard Default response if requested in the ZCL header of the *PowerProfileResponse* command.

The device that receives the *PowerProfileResponse* command SHALL reply with a standard ZCL Default response with status UNSUP_CLUSTER_COMMAND 0x81 (as from ZCL specification) if the reception of this command is not supported.

If the profile data requested are not available, the device SHALL reply with a standard ZCL response INVALID_VALUE 0x87 (as ZCL specification).

3.17.6.3 PowerProfileStateResponse Command

The *PowerProfileStateResponse* command allows a device (server) to communicate its current Power Profile(s) to another device (client) that previously requested them.

3.17.6.3.1 Payload Format

The *PowerProfileStateResponse* command payload SHALL be formatted as illustrated in Figure 3-73.

Figure 3-73. Format of the *PowerProfileStateResponse* Command Frame

Octets	1	4	4	...	4
Field Name	Power Profile Count	Power Profile Record 1	Power Profile Record 2	...	Power Profile Record *n*

Each Power Profile record SHALL be formatted as illustrated in Figure 3-74.

Figure 3-74. Format of the Power Profile Record Field

Octets	1	1	1	1
Data Type	uint8	uint8	bool	enum8
Field Name	Power Profile ID	Energy Phase ID	PowerProfile RemoteControl	PowerProfile State

3.17.6.3.1.1 Payload Details

Power Profile Count

The Power Profile Count is the number of Power Profile Records that follow in the message.

Power Profile Record

The Power Profile record supports the following fields:

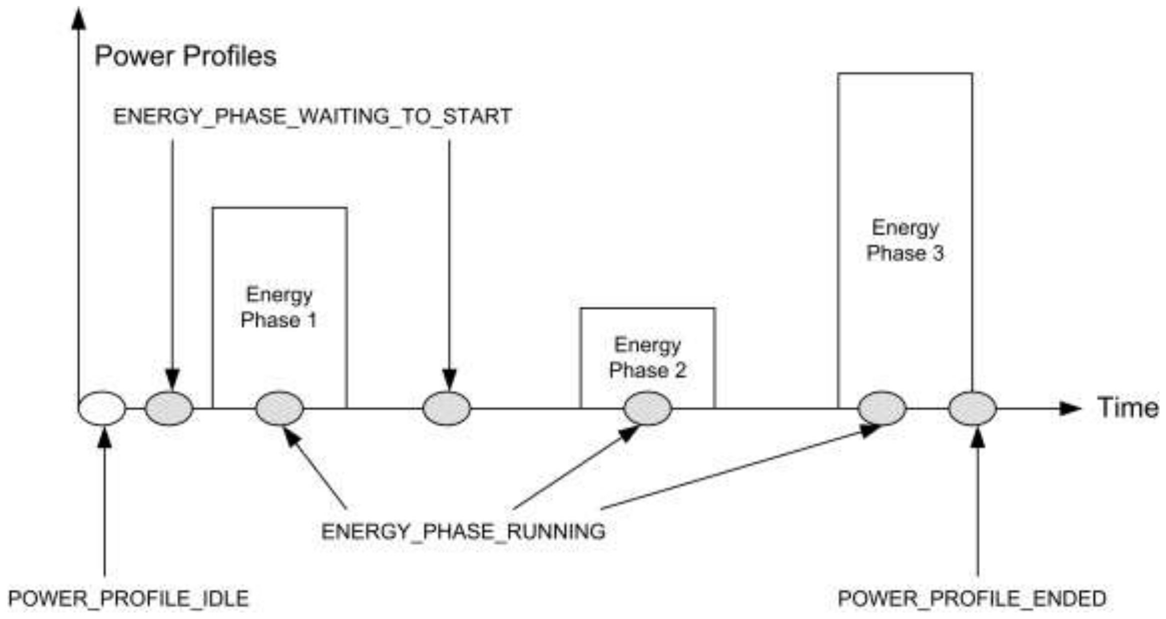
- **Power Profile ID:** The identifier of the Power Profile as requested.

- **Energy Phase ID:** The current Energy Phase ID of the specific Profile ID; this value SHALL be set to invalid 0xFF when PowerProfileState indicates a Power Profile in POWER_PROFILE_IDLE state.
- **PowerProfileRemoteControl:** It indicates if the PowerProfile is currently remotely controllable or not; if the Power Profile is not remotely controllable it cannot be scheduled by a Power Profile client.
- **PowerProfileState:** An enumeration field representing the current state of the Power Profile (see Table 3-140).

Table 3-140. *PowerProfileState* Enumeration Field

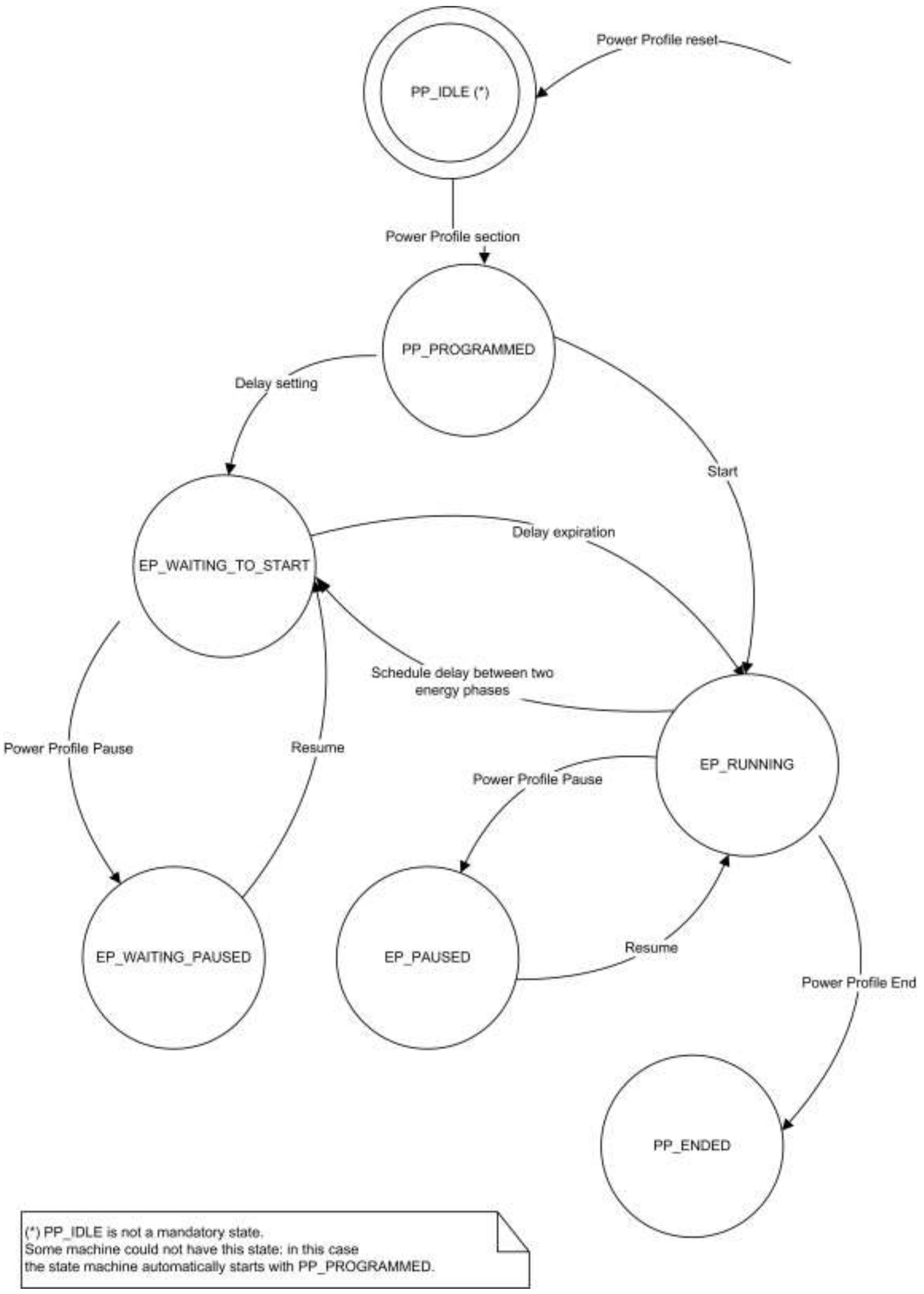
Enumeration	Value	Description
POWER_PROFILE_IDLE	0x00	The PP is not defined in its parameters.
POWER_PROFILE_PROGRAMMED	0x01	The PP is defined in its parameters but without a scheduled time reference
ENERGY_PHASE_RUNNING	0x03	An energy phase is running
ENERGY_PHASE_PAUSED	0x04	The current energy phase is paused
ENERGY_PHASE_WAITING_TO_START	0x05	The Power Profile is in between two energy phases (one ended, the other not yet started). If the first Energy Phase is considered, this state indicates that the whole power profile is not yet started, but it has been already programmed to start
ENERGY_PHASE_WAITING_PAUSED	0x06	The Power Profile is set to Pause when being in the ENERGY_PHASE_WAITING_TO_START state.
POWER_PROFILE_ENDED	0x07	The whole Power Profile is terminated

6570 **Figure 3-75. Power Profile States**



6571

6572 **Figure 3-76. Power Profile State Diagram**



6573

6574 3.17.6.3.2 When Generated

This command is generated when the command *PowerProfileStateRequest* is received. For more information, see Chapter 15, Appliance Management, section 3.

3.17.6.3.3 Effect on Receipt

On receipt of this command, the originator is notified of the results of its Read Current Power Profiles attempt (i.e., receives the Power Profiles currently running in the server device).

3.17.6.4 GetPowerProfilePrice Command

The *GetPowerProfilePrice* command is generated by the server (e.g., White Goods) in order to retrieve the cost associated to a specific Power Profile. This command has the same payload as the Power Profile Request command. For more information, see Chapter 15, Appliance Management, section 3.

3.17.6.4.1 Effect on Receipt

On receipt of this command, the recipient device SHALL generate a *GetPowerProfilePriceResponse* command. For more information, see Chapter 15, Appliance Management, section 3.

3.17.6.5 PowerProfileStateNotification Command

The *PowerProfileStateNotification* command is generated by the server (e.g., White Goods) in order to update the state of the power profile and the current energy phase. It has the same payload as the *PowerProfileStateResponse* command but it is an unsolicited command.

3.17.6.5.1 Effect on Receipt

On receipt of this command, the recipient device will update its information related to the PowerProfile of the device (e.g., it will update the forecasts of the durations of the Power Profile's energy phases with the actual data).

3.17.6.6 GetOverallSchedulePrice Command

The *GetOverallSchedulePrice* command is generated by the server (e.g., White Goods) in order to retrieve the overall cost associated to all the Power Profiles scheduled by the scheduler (the device supporting the Power Profile cluster client side) for the next 24 hours. This command has no payload.

3.17.6.6.1 Effect on Receipt

On receipt of this command, the recipient device SHALL generate a *GetOverallSchedulePriceResponse* command. For more information, see Chapter 15, Appliance Management, section 3.

3.17.6.7 EnergyPhasesScheduleRequest Command

The *EnergyPhasesScheduleRequest* Command is generated by the server (e.g., White Goods) in order to retrieve from the scheduler (e.g., Home Gateway) the schedule (if available) associated to the specific Power Profile carried in the payload. This command has the same payload as the Power Profile Request. For more information, see Chapter 15, Appliance Management, section 3.

3.17.6.7.1 Effect on Receipt

On receipt of this command, the recipient device SHALL generate an *EnergyPhasesScheduleResponse* command in order to notify the proper scheduling to the server side of the Power Profile cluster (“solicited” schedule). For more information, see Chapter 15, Appliance Management, section 3. If the schedule is accepted by the PowerProfile server side (e.g., the appliance) the PowerProfile SHALL have the state ENERGY_PHASE_WAITING_TO_START (delay start set for the first energy phase of the power profile). If the device receiving the *EnergyPhasesScheduleResponse* command cannot accept the schedule of the energy phases because the activation delay related to any of carried phases is equal to zero, it SHALL reply with a standard Default response with the error code NOT_AUTHORIZED (0x7e).

3.17.6.8 EnergyPhasesScheduleStateResponse Command

The *EnergyPhasesScheduleStateResponse* command is generated by the server (e.g., White Goods) in order to reply to an *EnergyPhasesScheduleStateRequest* command about the scheduling states that are set in the server side. For more information, see Chapter 15, Appliance Management, section 3. The payload of this command is the same as *EnergyPhasesScheduleNotification*. In case of no scheduled energy phases, the payload shown in Figure 3-77 SHALL be used.

Figure 3-77. Format of *EnergyPhasesScheduleStateResponse* in Case of No Scheduled Phases

Octets	1	1
Data Type	uint8	uint8
Field Name	PowerProfileID	Num of Scheduled Energy Phases=0x00

3.17.6.8.1 Effect on Receipt

On receipt of this command, the recipient device will be notified about the scheduling activity of the server side of the Power Profile Cluster.

Please note that the schedules MAY be set by the scheduling commands listed in this cluster or by the users (e.g., delay start of an appliance).

3.17.6.9 EnergyPhasesScheduleStateNotification Command

The *EnergyPhasesScheduleStateNotification* command is generated by the server (e.g., White Goods) in order to notify (un-solicited command) a client side about the scheduling states that are set in the server side. The payload of this command is the same as *EnergyPhasesScheduleStateResponse*.

3.17.6.9.1 Effect on Receipt

On receipt of this command, the recipient devices will be notified about the scheduling activity of the server side of the Power Profile Cluster.

3.17.6.10 PowerProfileScheduleConstraintsNotification Command

The *PowerProfileScheduleConstraintsNotification* command is generated by a device supporting the server side of the Power Profile cluster to notify the client side of this cluster about the imposed constraints and let the scheduler (i.e., the entity supporting the Power Profile cluster client side) to set the proper boundaries for the scheduling.

3.17.6.10.1 Payload Format

The *PowerProfileScheduleConstraintsNotification* command payload is reported in Figure 3-78.

Figure 3-78. Format of the *PowerProfileScheduleConstraintsNotification* Command Frame

Octets	1	2	2
Data Type	uint8	uint16	uint16
Field Name	PowerProfileID	Start After	Stop Before

3.17.6.10.1.1 Payload Details

The payload of the *PowerProfileScheduleConstraintsNotification* command carries the following fields:

- The Power Profile ID field specifies which profile (among *TotalProfileNum* total profiles number) the constraints are referring to.
- The *StartAfter* parameter represents the relative time in minutes (in respect to the time of the reception of this command), that limits the start of the Power Profile; it means that the Power Profile SHOULD be scheduled to start after a period of time equal to *StartAfter*; if this value is not specified by the device the value SHALL be 0x0000;
- The *StopBefore* parameter represents the relative time in minutes (in respect to the time of the reception of this command), that limits the end of the Power Profile; it means that the Power Profile SHOULD be scheduled to end before a period of time equal to *StopBefore*; if this value is not specified by the device the value SHALL be 0xFFFF.

3.17.6.10.2 When Generated

This command is generated when the server side of the Power Profile cluster (e.g., a White Goods device), needs to notify a change in the constraints of the Power Profile (e.g., the user selected boundaries for the specific behavior of the device).

3.17.6.10.3 Effect on Receipt

The device that receives the *PowerProfileScheduleConstraintsNotification* command SHALL use the information carried in the payload of this command to refine the proper schedule of the specific Power Profile indicated in the Power Profile ID field in order to meet the constraints.

3.17.6.11 PowerProfileScheduleConstraintsResponse Command

The *PowerProfileScheduleConstraintsResponse* command is generated by a device supporting the server side of the Power Profile cluster to reply to a client side of this cluster which sent a *PowerProfileScheduleConstraintsRequest*. The payload carries the selected constraints to let the scheduler (i.e., the entity supporting the Power Profile client cluster) to set the proper boundaries for completing or refining the scheduling.

3.17.6.11.1 Payload Format

Same as *PowerProfileScheduleConstraintsNotification* command. For more information, see Chapter 15, Appliance Management, section 3.

3.17.6.11.2 When Generated

This command is generated as a reply to the Power Profile Schedule Constraints Request.

3.17.6.11.3 Effect on Receipt

The device that receives the Power Profile Schedule ConstraintsResponse command SHALL use the information carried in the payload of this command to refine the proper schedule of the specific Power Profile indicated in the Power ProfileID field.

3.17.6.12 GetPowerProfilePriceExtended Command

The *GetPowerProfilePriceExtended* command is generated by the server (e.g., White Goods) in order to retrieve the cost associated to a specific Power Profile considering specific conditions described in the option field (e.g., a specific time).

3.17.6.12.1 Payload Format

The *GetPowerProfilePriceExtended* command payload SHALL be formatted as illustrated in Figure 3-79.

Figure 3-79. Format of the *GetPowerProfilePriceExtended* Command Payload

Octets	1	1	0/2
Data Type	map8	uint8	uint16
Field Name	Options	PowerProfileID	PowerProfileStartTime

Table 3-141. Options Field

Bit	**Description**
0	Bit0=1 : PowerProfileStartTime Field Present
1	Bit1=0 : provide an estimation of the price considering the power profile with contiguous energy phases Bit1=1 : provide an estimation of the price considering the power profile as scheduled (i.e., taking in account delays between Energy phases set by the EMS)

Options

The Options field represents the type of request of extended price is requested to the client side of the power profile cluster (e.g., to an energy management system).

PowerProfileStartTime

The PowerProfileStartTime field represents the relative time (expressed in relative encoding in respect to the current time) when the overall Power Profile can potentially start. The unit is the minute.

3.17.6.12.2 Effect on Receipt

On receipt of this command, the recipient device SHALL generate a *GetPowerProfilePriceExtendedResponse* command. For more information, see Chapter 15, Appliance Management, section 3.

3.17.7 Client Attributes

The client has no cluster specific attributes.

3.17.8 Client Commands Received

Description is in server side commands generated (sent) description.

3.17.9 Client Commands Generated

Description is in server side commands received description.

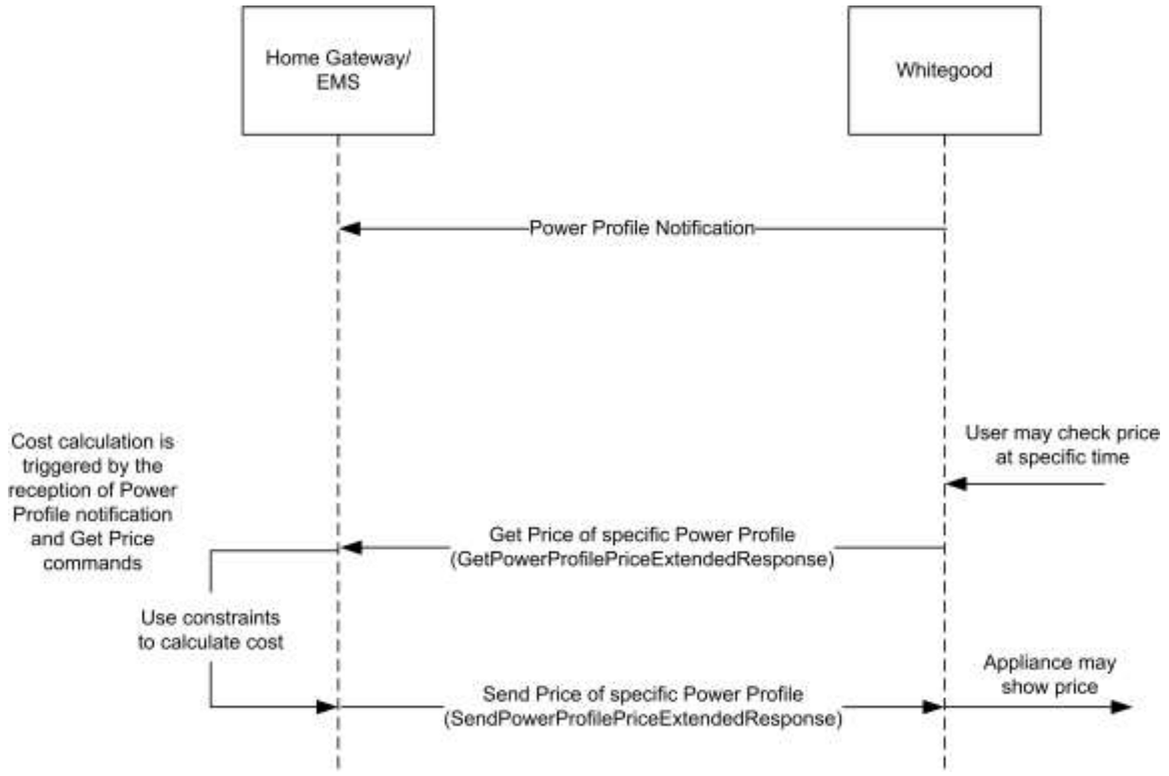
3.17.10 Example of Device Interactions Using the Power Profile (Informative Section)

3.17.10.1 Price Information Retrieved by the White Goods

The price of a specific appliance program is estimated by the Home gateway/EMS, calculated using the Power Profile forecast provided by the appliance and the *PowerProfileStartTime* contained in the *GetPowerProfilePriceExtended* which indicates when the appliance program will start. How the Home Gateway/EMS retrieves from the utility the information related to tariff schemes and price changes over time is out of scope of this specification.

The appliance MAY then show to the user on the display the price associated to a specific cycle set (e.g., a washing machine program "Cotton 90 °C" will cost you "1.15€").

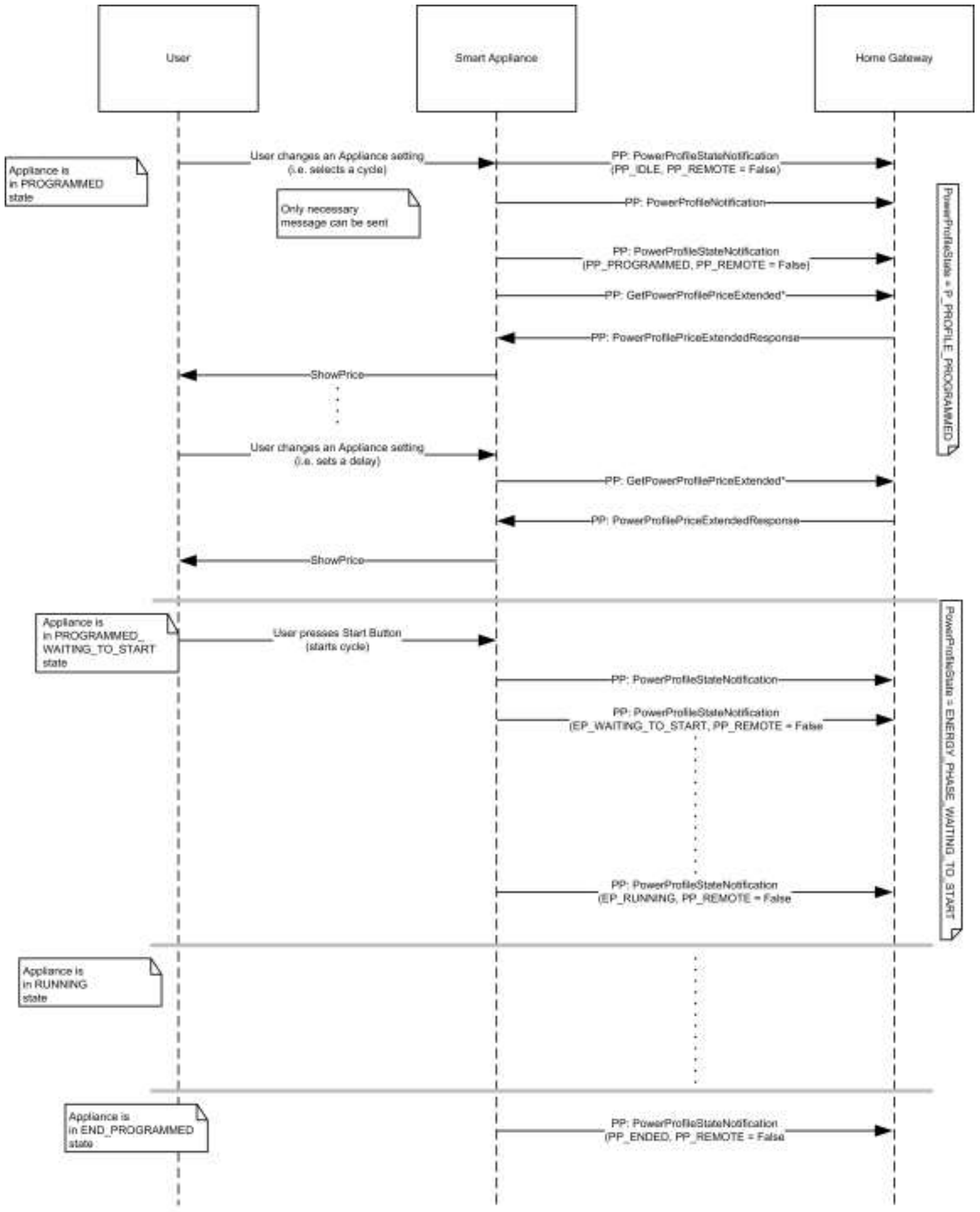
6710 **Figure 3-80. Visualization of Price Associated to a Power Profile**


Home Gateway/ EMS
Whitegood
Power Profile Notification
User may check price at specific time
Cost calculation is triggered by the reception of Power Profile notification and Get Price commands
Get Price of specific Power Profile (GetPowerProfilePriceExtendedResponse)
Use constraints to calculate cost
Appliance may show price
Send Price of specific Power Profile (SendPowerProfilePriceExtendedResponse)

6711

3.17.10.2 Interaction with Power Profile Cluster When Appliance Is Not Remotely Controllable

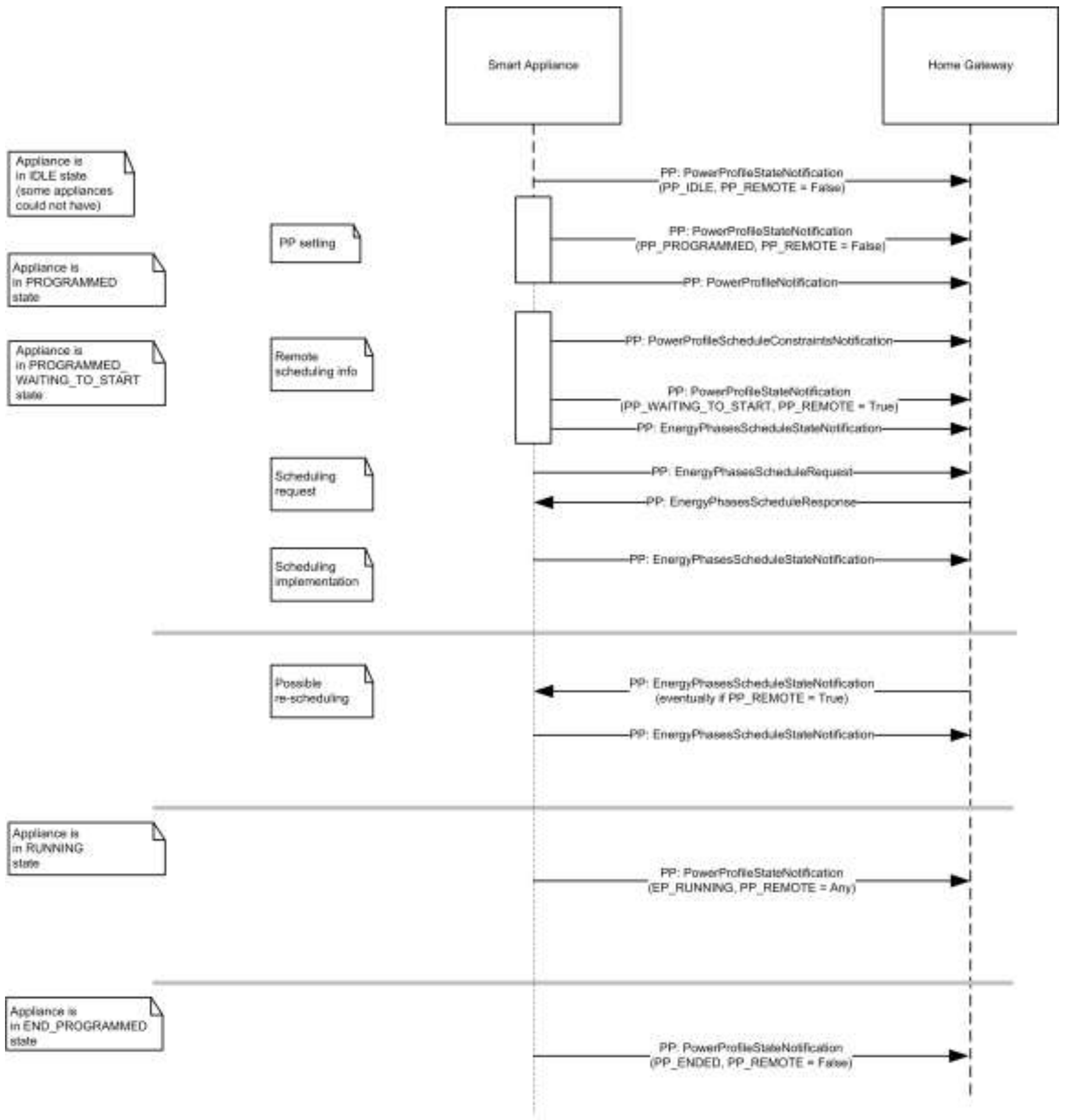
Figure 3-81. Energy Remote Disabled: Example of Sequence Diagram with User Interaction


User
Smart Appliance
Home Gateway
Appliance is in PROGRAMMED state
User changes an Appliance setting (i.e. selects a cycle)
PP: PowerProfileStateNotification (PP_IDLE, PP_REMOTE = False)
Only necessary message can be sent
PP: PowerProfileNotification
PP: PowerProfileStateNotification (PP_PROGRAMMED, PP_REMOTE = False)
PP: GetPowerProfilePriceExtended*
PP: PowerProfilePriceExtendedResponse
ShowPrice
User changes an Appliance setting (i.e. sets a delay)
PP: GetPowerProfilePriceExtended*
PP: PowerProfilePriceExtendedResponse
ShowPrice
PowerProfileState = P_PROFILE_PROGRAMMED
Appliance is in PROGRAMMED_WAITING_TO_START state
User presses Start Button (starts cycle)
PP: PowerProfileStateNotification
PP: PowerProfileStateNotification (EP_WAITING_TO_START, PP_REMOTE = False
PP: PowerProfileStateNotification (EP_RUNNING, PP_REMOTE = False
PowerProfileState = ENERGY_PHASE_WAITING_TO_START
Appliance is in RUNNING state
Appliance is in END_PROGRAMMED state
PP: PowerProfileStateNotification (PP_ENDED, PP_REMOTE = False

*GetPowerProfilePriceExtended payload includes delay time to start

3.17.10.3 Interaction with Power Profile Cluster When Appliance Is Remotely Controllable (Scheduling of Appliance)

Figure 3-82. Energy Remote Enabled: Example of Sequence Diagram with User Interaction


Smart Appliance
Home Gateway
Appliance is in IDLE state (some appliances could not have)
PP: PowerProfileStateNotification (PP_IDLE, PP_REMOTE = False)
PP setting
PP: PowerProfileStateNotification (PP_PROGRAMMED, PP_REMOTE = False)
Appliance is in PROGRAMMED state
PP: PowerProfileNotification
PP: PowerProfileScheduleConstraintsNotification
Appliance is in PROGRAMMED_ WAITING_TO_START state
Remote scheduling info
PP: PowerProfileStateNotification (PP_WAITING_TO_START, PP_REMOTE = True)
PP: EnergyPhasesScheduleStateNotification
Scheduling request
PP: EnergyPhasesScheduleRequest
PP: EnergyPhasesScheduleResponse
Scheduling implementation
PP: EnergyPhasesScheduleStateNotification
Possible re-scheduling
PP: EnergyPhasesScheduleStateNotification (eventually if PP_REMOTE = True)
PP: EnergyPhasesScheduleStateNotification
Appliance is in RUNNING state
PP: PowerProfileStateNotification (EP_RUNNING, PP_REMOTE = Any)
Appliance is in END_PROGRAMMED state
PP: PowerProfileStateNotification (PP_ENDED, PP_REMOTE = False)

*GetPowerProfilePriceExtended can be generated any time by SA if a PP is active

3.18 Meter Identification

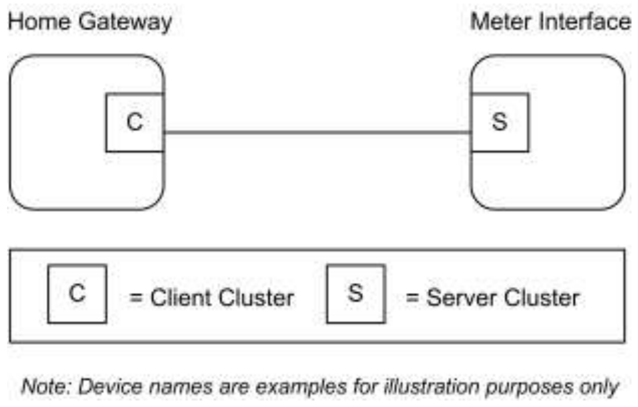
3.18.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides attributes and commands for determining advanced information about utility metering device, as shown in Figure 3-83.

Note: *Where a physical node supports multiple endpoints it will often be the case that many of these settings will apply to the whole node, that is they are the same for every endpoint on the device. In such cases they can be implemented once for the node, and mapped to each endpoint.*

Figure 3-83. Typical Usage of the Meter Identification Cluster



3.18.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added

3.18.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MTRID	Type 2 (server to client)

3.18.1.3 Cluster Identifiers

Identifier	Name
0x0b01	Meter Identification

3.18.2 Server

3.18.2.1 Meter Identification Attribute Set

The Meter Identification server cluster contains the attributes summarized in Table 3-142.

Table 3-142. Attributes of the Meter Identification Server Cluster

Identifier	Name	Type	Range	Access	Def	M/O
0x0000	*CompanyName*	string	0 to 16 Octets	R	-	M
0x0001	*MeterTypeID*	uint16	0x0000 to 0xffff	R	-	M
0x0004	*DataQualityID*	uint16	0x0000 to 0xffff	R	-	M
0x0005	*CustomerName*	string	0 to 16 Octets	RW	-	O
0x0006	*Model*	octstr	0 to 16 Octets	R	-	O
0x0007	*PartNumber*	octstr	0 to 16 Octets	R	-	O
0x0008	*ProductRevision*	octstr	0 to 6 Octets	R	-	O
0x000A	*SoftwareRevision*	octstr	0 to 6 Octets	R	-	O
0x000B	*UtilityName*	string	0 to 16 Octets	R	-	O
0x000C	*POD*	string	0 to 16 Octets	R	-	M
0x000D	*AvailablePower*	int24	0x000000 to 0xffffff	R	-	M
0x000E	*PowerThreshold*	int24	0x000000 to 0xffffff	R	-	M

3.18.2.1.1 CompanyName Attribute

CompanyName is a ZCL Octet String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the UTF-8 format. Company Name defines the meter manufacturer name, decided by manufacturer.

3.18.2.1.2 MeterTypeID Attribute

MeterTypeID defines the Meter installation features, decided by manufacturer. Table 3-143 provides Meter Type IDs field content.

Table 3-143. Meter Type IDs

Device	Meter Type ID
Utility Primary Meter	0x0000
Utility Production Meter	0x0001
Utility Secondary Meter	0x0002
Private Primary Meter	0x0100
Private Production Meter	0x0101
Private Secondary Meters	0x0102
Generic Meter	0x0110

3.18.2.1.3 DataQualityID Attribute

DataQualityID defines the Meter Simple Metering information certification type, decided by manufacturer.

Table 3-144 provides Data Quality IDs field content.

Table 3-144. Data Quality IDs

Device	Meter Type ID
All Data Certified	0x0000
Only Instantaneous Power not Certified	0x0001
Only Cumulated Consumption not Certified	0x0002
Not Certified data	0x0003

3.18.2.1.4 CustomerName Attribute

CustomerName is a ZCL Character String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the ASCII format.

3.18.2.1.5 *Model* Attribute

Model is a ZCL Octet String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the UTF-8 format. *Model* defines the meter model name, decided by manufacturer.

3.18.2.1.6 *PartNumber* Attribute

PartNumber is a ZCL Octet String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the UTF-8 format. *PartNumber* defines the meter part number, decided by manufacturer.

3.18.2.1.7 ProductRevision Attribute

ProductRevision is a ZCL Octet String field capable of storing up to 6 character string (the first Octet indicates length) encoded in the UTF-8 format. *ProductRevision* defines the meter revision code, decided by manufacturer.

3.18.2.1.8 SoftwareRevision Attribute

SoftwareRevision is a ZCL Octet String field capable of storing up to 6 character string (the first Octet indicates length) encoded in the UTF-8 format. *SoftwareRevision* defines the meter software revision code, decided by manufacturer.

3.18.2.1.9 UtilityName Attribute

UtilityName is a ZCL Character String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the ASCII format.

3.18.2.1.10 *POD* Attribute

POD (Point of Delivery) is a ZCL Character String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the ASCII format. POD is the unique identification ID of the premise connection point. It is also a contractual information known by the clients and indicated in the bill.

3.18.2.1.11 AvailablePower Attribute

AvailablePower represents the *InstantaneousDemand* that can be distributed to the customer (e.g., 3.3KW power) without any risk of overload. The *Available Power* SHALL use the same formatting conventions as the one used in the simple metering cluster formatting attribute set for the *InstantaneousDemand* attribute, i.e., the *UnitOfMeasure* and *DemandFormatting*.

3.18.2.1.12 PowerThreshold Attribute

PowerThreshold represents a threshold of *InstantaneousDemand* distributed to the customer (e.g., 4.191KW) that will lead to an imminent risk of overload. The *PowerThreshold* SHALL use the same formatting conventions as the one used in the *AvailablePower* attributes and therefore in the simple metering cluster formatting attribute set for the *InstantaneousDemand* attribute, i.e., the *UnitOfMeasure* and *DemandFormatting*.

3.18.2.1.13 Commands Received

No cluster-specific commands are received or generated by the server.

3.18.3 Client

The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

3.19 Level Control for Lighting

3.19.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface for controlling the level of a light source.

All requirements and dependencies are derived from the base cluster. Additional or extended requirements and dependencies are listed in this cluster specification.

NOTE: This cluster specification is derived from the Level cluster specification (generic level control also defined in this document). This cluster specifies further requirements for the lighting application. Please see section 3.10 for the generic Level cluster specification.

3.19.1.1 Revision History

Rev	**Description**
1	global mandatory *ClusterRevision* attribute added
2	added *Options* attribute, state change table; ZLO 1.0; Derived from base Level (no change)

	CCB 2085 1775 2281 2147

3.19.1.2 Classification

Hierarchy	Base	Role	PICS Code	Primary Transaction
Derived	Level (3.10)	Application	LC	Type 1 (client to server)

3.19.1.3 Cluster Identifiers

Identifier	Name
0x0008	Level Control for Lighting

3.19.2 Server

3.19.2.1 Dependencies

Please see examples of state changes with regards to the On/Off server cluster in section 3.19.4.

3.19.2.2 Attributes

The Level Control for Lighting server cluster supports the base attributes below. See the base cluster for details. Additional requirements are defined here.

Table 3-145. Attributes of the Level Control for Lighting server cluster

Name	Range	Default	M/O
CurrentLevel	0x01 to 0xfe	0xff	M
RemainingTime	*base*[43]	*base*	O
CurrentFrequency	*base*	*base*	O
MinFrequency	*base*	*base*	M:*CurrentFrequency*[44]
MaxFrequency	*base*	*base*	M:*CurrentFrequency*
OnOffTransitionTime	*base*	*base*	O
OnLevel	*base*	*base*	O
OnTransitionTime	*base*	*base*	O
OffTransitionTime	*base*	*base*	O

[43] see base cluster for definition
[44] Mandatory if *CurrentFrequency* supported

Name	Range	Default	M/O
DefaultMoveRate	*base*	*base*	O
Options	*base*	*base*	M
StartUpCurrentLevel	*base*	*base*	O

3.19.2.2.1 *CurrentLevel* Attribute for Lighting

A value of 0x00 SHALL not be used.

A value of 0x01 SHALL indicate the minimum level that can be attained on a device.

A value of 0xfe SHALL indicate the maximum 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 level.

3.19.2.2.2 CurrentFrequency Attribute

The *CurrentFrequency* attribute represents the frequency in 10Hz (hertz) increments up to 655.34 kHz. A value of 0 is unknown.

3.19.2.2.3 *Options* Attribute for Lighting

Below describes the lighting specific bits of the *Options* attribute. All other bits are defined, or reserved by the base cluster.

Table 3-146. *Options* Attribute

Bit	Name	Values & Summary
1	CoupleColorTempToLevel (See Color Control cluster)	0 - Do not couple changes to the *CurrentLevel* attribute with the color temperature. 1 - Couple changes to the *CurrentLevel* attribute with the color temperature.

3.19.2.3 Commands Received

The command IDs for the cluster are listed below:

Table 3-147. Commands for the Pulse Width Modulation cluster

ID	Description	M/O
0x00	Move to Level	M
0x01	Move	M

ID	Description	M/O
0x02	Step	M
0x03	Stop	M
0x04	Move to Level (with On/Off)	M
0x05	Move (with On/Off)	M
0x06	Step (with On/Off)	M
0x07	Stop	M
0x08	Move to Closest Frequency	M:*CurrentFrequency*

3.19.2.4 Commands Generated

There are no commands generated by the server.

3.19.3 Client

The client has no cluster specific attributes. The client generates the cluster specific commands received by the server, as required by the application. No cluster specific commands are received by the client.

3.19.4 State Change Table for Lighting

Below is a table of examples of state changes when Level Control for Lighting and On/Off clusters are on the same endpoint.

EiO - *ExecuteIfOff* field in the *Options* attribute

OnOff – attribute value of On/Off cluster 0=Off, 1=On

MIN - *MinLevel*

MAX - *MaxLevel*

MID – midpoint between *MinLevel* and *MaxLevel*

Table 3-148. Lighting Device State Change

Current Level	***EiO***	***OnOff***	**Physical Device**	**Command ← Before After →**	***Current Level***	***OnOff***	**Physical Device**	**Device Output Result**
any	0	0	Off	Move to level *MID* over 2 sec	*same*	0	Off	stays off
any	0	0	Off	Move with On/Off to level *MID* over 2 sec	*MID*	1	On (mid-point brightness)	turns on and output level adjusts or stays at half

Current Level	***EiO***	***OnOff***	**Physical Device**	**Command** ← Before After →	***Current Level***	***OnOff***	**Physical Device**	**Device Output Result**
any	1	0	Off	Move to level *MID* over 2 sec	*MID*	0	Off	stays off
any	1	0	Off	Move with On/Off to level *MID* over 2 sec	*MID*	1	On	turns on and output level adjusts to or stays at half
any	1	0	Off	Move rate = up 64 per second	*MAX*	0	Off	stays off
any	1	0	Off	Move with On/Off rate = up 64 per second	*MAX*	1	On	turn on and output level adjusts to or stays at full
any	1	0	Off	Move (with On/Off) rate = down 64 per second	*MIN*	0	Off	stays off
any	*any*	1	On (any brightness)	Move (with On/Off) to level *MID* over 2 sec	*MID*	1	On (mid-point brightness)	output level adjusts to or stays at half
any	*any*	1	On (any brightness)	Move (with On/Off) rate = up 64 per second	*MAX*	1	On (full brightness)	output level adjusts to or stays at full
any	*any*	1	On (any brightness)	Move rate = down 64 per second	*MIN*	0	On (at minimum brightness)	output level adjusts to minimum
any	*any*	1	On (any brightness)	Move with On/Off rate = down 64 per second	*MIN*	0	Off	output level adjusts to off

6842

3.20 Pulse Width Modulation

3.20.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface for controlling the Pulse Width Modulation (PWM) characteristics of a device. Typical applications include heating element and fan control (10-100Hz), DC electric motors and power efficient LED control (5-10kHz), and switching power supplies (>20kHz). For the purposes of PWM, the value of level is effectively a duty cycle. The frequency and level (duty cycle) values are reportable and may be configured for reporting.

3.20.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added

3.20.1.2 Classification

Hierarchy	Base	Role	PICS Code	Primary Transaction
Derived	Level (3.10)	Application	PWM	Type 1 (client to server)

3.20.1.3 Cluster Identifiers

Identifier	Name
0x001c	Pulse Width Modulation

3.20.2 Server

The server requirements and dependencies are derived from the base cluster. Additional requirements are listed in this section below.

3.20.2.1 Attributes

The Pulse Width Modulation server cluster supports the base attributes below. See the base cluster for details. Additional requirements are defined here.

Table 3-149. Attributes of the Pulse Width Modulation server cluster

Name	Range	Default	M/O
CurrentLevel	*MinLevel* to *MaxLevel*	0xff	M
MinLevel	0 to *MaxLevel*	0	M
MaxLevel	*MinLevel* to 100	100	M
CurrentFrequency	*MinFrequency* to *MaxFrequency*	0	M

Name	Range	Default	M/O
MinFrequency	0 to *MaxFrequency*	0	M
MaxFrequency	*MinFrequency* to 0xffff	0	M

3.20.2.1.1 CurrentLevel Attribute

The *CurrentLevel* attribute represents a duty cycle as a percentage. A value of 0xff is unknown.

3.20.2.1.2 CurrentFrequency Attribute

The *CurrentFrequency* attribute represents the frequency in 10Hz (hertz) increments up to 655.34 kHz. A value of 0 is unknown.

3.20.2.2 Commands Received

The command IDs for the cluster are listed below:

Table 3-150. Commands for the Pulse Width Modulation cluster

ID	Description	M/O
0x00	Move to Level	M
0x01	Move	M
0x02	Step	M
0x03	Stop	M
0x04	Move to Level (with On/Off)	M
0x05	Move (with On/Off)	M
0x06	Step (with On/Off)	M
0x07	Stop	M
0x08	Move to Closest Frequency	M

3.20.2.3 Commands Generated

There are no commands generated by the server.

3.20.3 Client

The client has no cluster specific attributes. The client generates the cluster specific commands received by the server, as required by the application. No cluster specific commands are received by the client.

CHAPTER 4 MEASUREMENT AND SENSING

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

4.1 General Description

4.1.1 Introduction

The clusters specified in this document are generic measurement and sensing interfaces that are sufficiently general to be of use across a wide range of application domains.

4.1.2 Cluster List

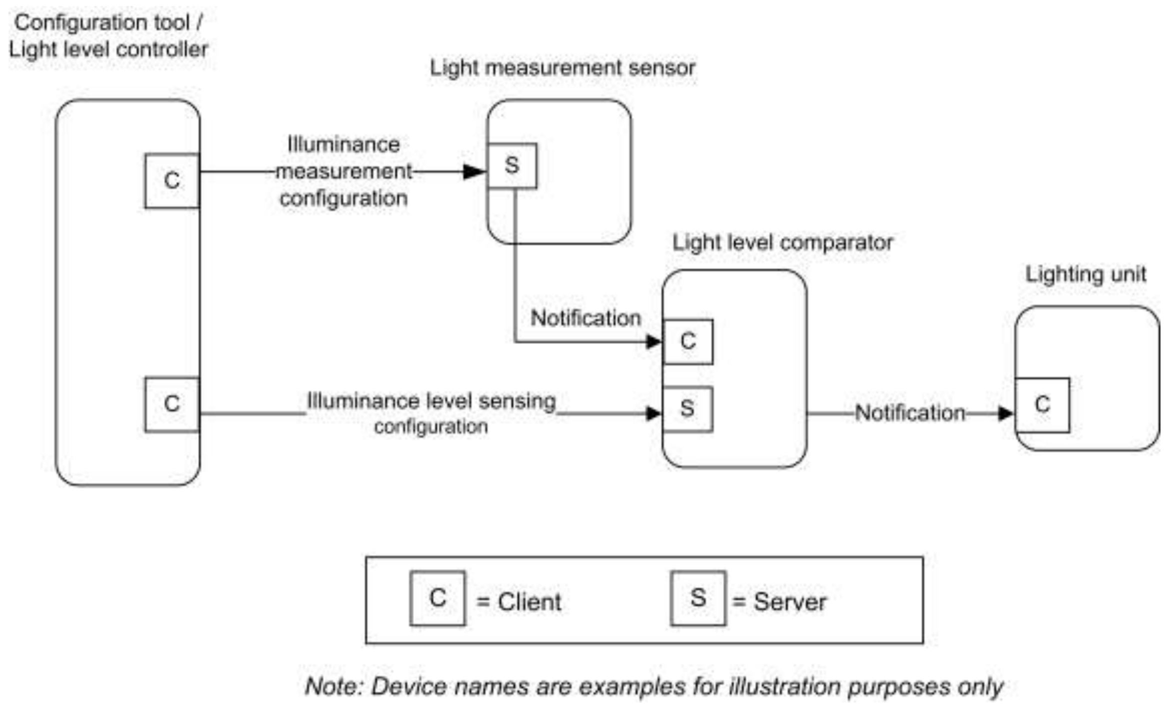
This section lists the clusters specified in this document, and gives examples of typical usage.

4.1.2.1 Illuminance Measurement and Level Sensing

Table 4-1. Illuminance Measurement and Level Sensing Clusters

Cluster ID	Cluster Name	Description
0x0400	Illuminance Measurement	Attributes and commands for configuring the measurement of illuminance, and reporting illuminance measurements
0x0401	Illuminance Level Sensing	Attributes and commands for configuring the sensing of illuminance levels, and reporting whether illuminance is above, below, or on target

6888 **Figure 4-1. Typical Usage of Illuminance Measurement and Level Sensing Clusters**


Configuration tool / Light level controller
Light measurement sensor
Illuminance measurement configuration
C
S
Light level comparator
Notification
Lighting unit
C
C
Illuminance level sensing configuration
S
Notification
C
C = Client
S = Server
Note: Device names are examples for illustration purposes only

6889

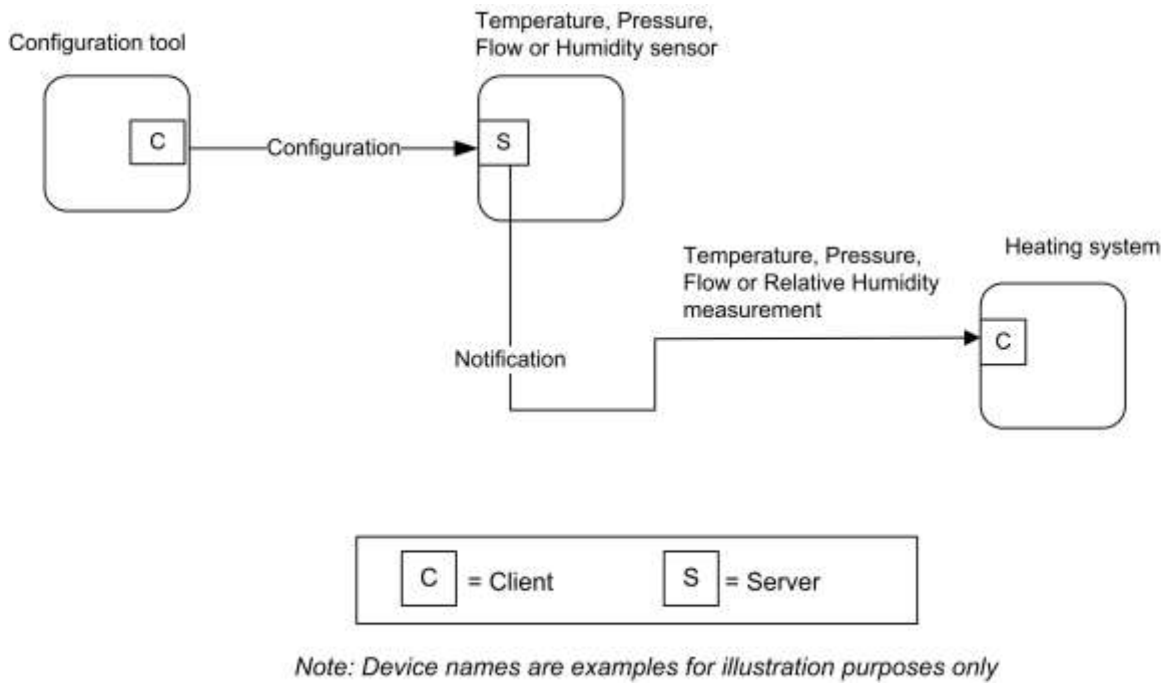
6890
4.1.2.2 Temperature, Pressure and Flow Measurement

6891 **Table 4-2. Pressure and Flow Measurement Clusters**

ID	Cluster Name	Description
0x0402	Temperature Measurement	Attributes and commands for configuring the measurement of temperature, and reporting temperature measurements
0x0403	Pressure Measurement	Attributes and commands for configuring the measurement of pressure, and reporting pressure measurements
0x0404	Flow Measurement	Attributes and commands for configuring the measurement of flow, and reporting flow rates
0x0405	Relative Humidity Measurement	Attributes and commands for configuring the measurement of relative humidity, and reporting relative humidity measurements

6892

6893 **Figure 4-2. Typical Usage of Temperature, Pressure and Flow Measurement Clusters**


Configuration tool
C
Configuration
Temperature, Pressure, Flow or Humidity sensor
S
Notification
Temperature, Pressure, Flow or Relative Humidity measurement
Heating system
C
C = Client
S = Server
Note: Device names are examples for illustration purposes only

6894

6895

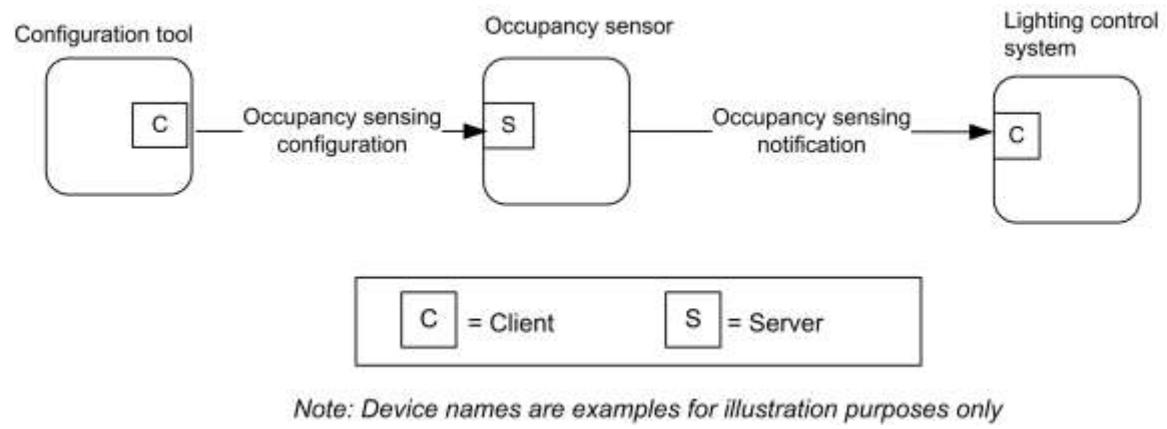
4.1.2.3 Occupancy Sensing

6896 **Table 4-3. Occupancy Sensing Clusters**

ID	Cluster Name	Description
0x0406	Occupancy Sensing	Attributes and commands for configuring occupancy sensing, and reporting occupancy status

6897

Figure 4-3. Typical Usage of Occupancy Sensing Cluster


Configuration tool
C
Occupancy sensing configuration
Occupancy sensor
S
Occupancy sensing notification
Lighting control system
C
C = Client
S = Server
Note: Device names are examples for illustration purposes only

4.1.2.4 Electrical Measurement

Table 4-4. Electrical Measurement Clusters

Cluster ID	Cluster Name	Description
0x0b04	Electrical Measurement	Attributes and commands for measuring electrical usage

4.1.3 Measured Value[45]

4.1.3.1 Range

For any measurement cluster with *MeasuredValue*, *MinMeasuredValue* and *MaxMeasuredValue* attributes, the following SHALL be always be true:

- *If both are defined, then MaxMeasuredValue* SHALL be greater than *MinMeasuredValue*.
- If *MaxMeasuredValue* is known, then *MeasuredValue* SHALL be less than or equal to *MaxMeasuredValue*.
- If *MinMeasuredValue* is known, then *MeasuredValue* SHALL be greater than or equal to *MinMeasuredValue*.

4.1.3.2 Tolerance

For any measurement cluster with a *MeasuredValue* and *Tolerance* attribute, the following SHALL always be true:

> The *Tolerance* attribute SHALL indicate the magnitude of the possible error that is associated with *MeasuredValue, using the same units and resolution.* The true value SHALL be in the range (*MeasuredValue* – *Tolerance*) to (*MeasuredValue* + *Tolerance*).

If known, the true value SHALL never be outside the possible physical range. Some examples:

- a temperature SHALL NOT be below absolute zero
- a concentration SHALL NOT be negative

[45] NFR Quality of Goods

4.2 Illuminance Measurement

4.2.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The server cluster provides an interface to illuminance measurement functionality, including configuration and provision of notifications of illuminance measurements.

4.2.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added; CCB 2048 2049 2050
2	CCB 2167

4.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	ILL	Type 2 (server to client)

4.2.1.3 Cluster Identifiers

Identifier	Name
0x0400	Illuminance Measurement

4.2.2 Server

4.2.2.1 Attributes

The Illuminance Measurement attributes summarized in Table 4-5.

Table 4-5. Illuminance Measurement Attributes

Id	Name	Type	Range	Acc	Def	MO
0x0000	*MeasuredValue*	uint16	*0x0000 to 0xffff*	RP	0x0000	M
0x0001	*MinMeasuredValue*	uint16	0x0001 – 0xfffd	R[46]	ms	M
0x0002	*MaxMeasuredValue*	uint16	*2* to 0xfffe	R	ms	M
0x0003	*Tolerance*	uint16	0x0000 – 0x0800	R	ms	O
0x0004	*LightSensorType*	enum8	0x00 – 0xff	R	0xff	O

[46] CCB 2167 typo: *MeasuredValue* should be reportable (not *MinMeasuredValue*)

4.2.2.1.1 *MeasuredValue* Attribute

MeasuredValue represents the Illuminance in Lux (symbol lx) as follows:

MeasuredValue = 10,000 x $\log_{10}$ Illuminance + 1

Where 1 lx <= Illuminance <=3.576 Mlx, corresponding to a *MeasuredValue* in the range 1 to 0xfffe.

The *MeasuredValue* attribute can take the following values.

- 0x0000 indicates a value of Illuminance that is too low to be measured.
- *MinMeasuredValue* ≤ *MeasuredValue* ≤ *MaxMeasuredValue under normal circumstances.*
- 0xffff indicates that the Illuminance measurement is invalid.

MeasuredValue is updated continuously as new measurements are made.

4.2.2.1.2 *MinMeasuredValue* Attribute

The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that can be measured. A value of 0xffff indicates that this attribute is not defined. See 4.1.3 for more details.

4.2.2.1.3 *MaxMeasuredValue* Attribute

The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that can be measured. A value of 0xffff indicates that this attribute is not defined. See 4.1.3 for more details.

4.2.2.1.4 *Tolerance* Attribute

See 4.1.3.

4.2.2.1.5 *LightSensorType* Attribute

The *LightSensorType* attribute specifies the electronic type of the light sensor. This attribute shall be set to one of the non-reserved values listed in Table 4-6.

Table 4-6. Values of the *LightSensorType* Attribute

Attribute Value	Description
0x00	Photodiode
0x01	CMOS
0x40 – 0xfe	Reserved for manufacturer specific light sensor types
0xff	Unknown

4.2.2.2 Commands

No cluster specific commands are generated or received by the server cluster.

4.2.2.3 Attribute Reporting

This cluster shall support attribute reporting using the Report Attributes command and according to the minimum and maximum reporting intervals and reportable change settings described in the ZCL Foundation specification (see 2.4.7). The following attributes shall be reported:

MeasuredValue

4.2.3 Client

The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

4.3 Illuminance Level Sensing

4.3.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The server cluster provides an interface to illuminance level sensing functionality, including configuration and provision of notifications of whether the illuminance is within, above or below a target band.

4.3.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

4.3.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	ILLVL	Type 2 (server to client)

4.3.1.3 Cluster Identifiers

Identifier	Name
0x0401	Illuminance Level Sensing

4.3.2 Server

4.3.2.1 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 4-7.

Table 4-7. Illuminance Level Sensing Attribute Sets

Attribute Set Identifier	Description
0x000	Illuminance Level Sensing Information
0x001	Illuminance Level Sensing Settings

4.3.2.2 Illuminance Level Sensing Information Attribute Set

The light sensor configuration attribute set contains the attributes summarized in Table 4-8.

Table 4-8. Illuminance Level Sensing Information Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0000	*LevelStatus*	enum8	0x00 – 0xfe	RP	-	M
0x0001	*LightSensorType*	enum8	0x00 – 0xfe	R	-	O

4.3.2.2.1 *LevelStatus* Attribute

The *LevelStatus* attribute indicates whether the measured illuminance is above, below, or within a band around *IlluminanceTargetLevel* (see 4.3.2.3.1). It may have any non-reserved value shown in Table 4-9.

Table 4-9. Values of the *LevelStatus* Attribute

Attribute Value	Description
0x00	Illuminance on target
0x01	Illuminance below target
0x02	Illuminance above target

4.3.2.2.2 *LightSensorType* Attribute

The *LightSensorType* attribute specifies the electronic type of the light sensor. This attribute shall be set to one of the non-reserved values listed in Table 4-10.

Table 4-10. Values of the *LightSensorType* Attribute

Attribute Value	Description
0x00	Photodiode
0x01	CMOS
0x40 – 0xfe	Reserved for manufacturer specific light sensor types
0xff	Unknown

4.3.2.3 Illuminance Level Sensing Settings Attribute Set

The light sensor configuration attribute set contains the attributes summarized in Table 4-11Illuminance Level Sensing Settings Attribute Set.

Table 4-11. Illuminance Level Sensing Settings Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0010	*IlluminanceTargetLevel*	uint16	0x0000 – 0xfffe	RW	-	M

4.3.2.3.1 *IlluminanceTargetLevel* Attribute

The *IlluminanceTargetLevel* attribute specifies the target illuminance level. This target level is taken as the centre of a 'dead band', which must be sufficient in width, with hysteresis bands at both top and bottom, to provide reliable notifications without 'chatter'. Such a dead band and hysteresis bands must be provided by any implementation of this cluster. (N.B. Manufacturer specific attributes may be provided to configure these).

IlluminanceTargetLevel represents illuminance in Lux (symbol lx) as follows:

IlluminanceTargetLevel = 10,000 x $\log_{10}$ Illuminance

Where 1 lx <= Illuminance <=3.576 Mlx, corresponding to a *MeasuredValue* in the range 0 to 0xfffe.

A value of 0xffff indicates that this attribute is not valid.

4.3.2.4 Commands Received

No cluster specific commands are received by the server.

4.3.2.5 Commands Generated

No cluster specific commands are generated by the server cluster.

4.3.2.6 Attribute Reporting

This cluster shall support attribute reporting using the Report Attributes command and according to the minimum and maximum reporting interval settings described in the ZCL Foundation Specification (see 2.4.7). The following attribute shall be reported:

LevelStatus

4.3.3 Client

The client cluster has no dependencies, specific attributes nor specific commands generated or received.

4.4 Temperature Measurement

4.4.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The server cluster provides an interface to temperature measurement functionality, including configuration and provision of notifications of temperature measurements.

4.4.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added
2	CCB 2241 2370

4.4.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	TMP	Type 2 (server to client)

4.4.1.3 Cluster Identifiers

Identifier	Name
0x0402	Temperature Measurement

4.4.2 Server

4.4.2.1 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant nibble specifies the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 4-12.

Table 4-12. Temperature Measurement Attribute Sets

Attribute Set Identifier	Description
0x000	Temperature Measurement Information

4.4.2.1.1 Temperature Measurement Information Attribute Set

The Temperature Measurement Information attribute set contains the attributes summarized in Table 4-13.

Table 4-13. Temperature Measurement Information Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*MeasuredValue*	int16	*MinMeasuredValue* – *MaxMeasuredValue*	RP	0xffff[47]	M
0x0001	*MinMeasuredValue*	int16	0x954d – 0x7ffe	R	0x8000	M
0x0002	*MaxMeasuredValue*	int16	0x954e – 0x7fff	R	0x8000	M
0x0003	*Tolerance*	uint16	0x0000 – 0x0800	R[48]	-	O

4.4.2.1.1.1 *MeasuredValue* Attribute

MeasuredValue represents the temperature in degrees Celsius as follows:

MeasuredValue = 100 x temperature in degrees Celsius.

Where -273.15°C <= temperature <= 327.67 ºC, corresponding to a *MeasuredValue* in the range 0x954d to 0x7fff. The maximum resolution this format allows is 0.01 ºC.

A *MeasuredValue* of 0x8000 indicates that the temperature measurement is unknown, otherwise the range SHALL be as described in 4.1.3.

MeasuredValue is updated continuously as new measurements are made. *MinMeasuredValue* and *MaxMeasuredValue* define the range of the sensor.

4.4.2.1.1.2 *MinMeasuredValue* Attribute

The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that is capable of being measured. A *MinMeasuredValue* of 0x8000 indicates that the minimum value is unknown. See 4.1.3 for more details.

4.4.2.1.1.3 *MaxMeasuredValue* Attribute

The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that is capable of being measured. See 4.1.3 for more details. A *MaxMeasuredValue* of 0x8000 indicates that the maximum value is unknown.

4.4.2.1.1.4 *Tolerance* Attribute

See 4.1.3.

4.4.2.2 Commands

No cluster specific commands are generated or received by the server cluster.

4.4.2.3 Attribute Reporting

This cluster shall support attribute reporting using the Report Attributes command and according to the minimum and maximum reporting interval and reportable change settings described in the ZCL Foundation specification (see 2.4.7). The following attributes shall be reported:

MeasuredValue [49]

[47] CCB 2370 define what a default value is
[48] CCB 2241 Tolerance is not required to be reportable if supported
[49] CCB 2241 Tolerance is not required to be reportable if supported

4.4.3 Client

The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

4.5 Pressure Measurement

4.5.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The server cluster provides an interface to pressure measurement functionality, including configuration and provision of notifications of pressure measurements.

4.5.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added
2	CCB 2241 2370

4.5.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	PRS	Type 2 (server to client)

4.5.1.3 Cluster Identifiers

Identifier	Name
0x0403	Pressure Measurement

4.5.2 Server

4.5.2.1 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 4-14Pressure Measurement Attribute Sets.

Table 4-14. Pressure Measurement Attribute Sets

Attribute Set Identifier	Description
0x000	Pressure Measurement Information

0x001	Extended Pressure Measurement Information

4.5.2.1.1 Pressure Measurement Information Attribute Set

The Pressure Measurement Information attribute set contains the attributes summarized in Table 4-15.

Table 4-15. Pressure Measurement Information Attribute Set

Id	Name	Type	Range	Acc	Def	MO
0x0000	*MeasuredValue*	int16	*MinMeasuredValue – MaxMeasuredValue*	RP	0x8000[50]	M
0x0001	*MinMeasuredValue*	int16	0x8001– 0x7ffe	R	0x8000	M
0x0002	*MaxMeasuredValue*	int16	0x8002– 0x7fff	R	0x8000	M
0x0003	*Tolerance*	uint16	0x0000 – 0x0800	R[51]	-	O

This set provides for measurements with a fixed maximum resolution of 0.1 kPa.

4.5.2.1.1.1 *MeasuredValue* Attribute

MeasuredValue represents the pressure in kPa as follows:

MeasuredValue = 10 x Pressure

Where -3276.7 kPa <= Pressure <= 3276.7 kPa, corresponding to a *MeasuredValue* in the range 0x8001 to 0x7fff.

MinMeasuredValue and *MaxMeasuredValue* define the range of the sensor.

A *MeasuredValue* of 0x8000 indicates that the pressure measurement is unknown, otherwise the range SHALL be as described in 4.1.3.

MeasuredValue is updated continuously as new measurements are made.

4.5.2.1.1.2 *MinMeasuredValue* Attribute

The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that can be measured. A value of 0x8000 means this attribute is not defined. See 4.1.3 for more details.

4.5.2.1.1.3 *MaxMeasuredValue* Attribute

The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that can be measured. A value of 0x8000 means this attribute is not defined. See 4.1.3 for more details.

4.5.2.1.1.4 *Tolerance* Attribute

See 4.1.3.

4.5.2.1.2 Extended Pressure Measurement Information Attribute Set

The Extended Pressure Measurement Information attribute set contains the attributes summarized in Table 4-16.

[50] CCB 2370 define default values
[51] CCB 2241 Tolerance is not required to be reportable if supported

Table 4-16. Extended Pressure Measurement Information Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0010	*ScaledValue*	int16	*MinScaledValue – MaxScaledValue*	R	0	O Note 1
0x0011	*MinScaledValue*	int16	0x8001-0x7ffe	R	0x8000	O Note 1
0x0012	*MaxScaledValue*	int16	0x8002-0x7fff	R	0x8000	O Note 1
0x0013	*ScaledTolerance*	uint16	0x0000 – 0x0800	R	-	O Note 2
0x0014	*Scale*	int8	0x81 – 0x7f	R	-	O Note 1

Note 1: If any one of these attributes is supported, all four shall be supported.
Note 2: If this attribute is supported, all attributes in this set shall be supported.

This attribute set is optional, and allows the range and resolution of measured pressures to be extended beyond those catered for by the Pressure Measurement Information Attribute Set, in a way fully backward compatible with devices that implement (or can read) only that attribute set.

4.5.2.1.2.1 *ScaledValue* Attribute

ScaledValue represents the pressure in Pascals as follows:

ScaledValue = 10^{Scale} x Pressure in Pa

Where -3276.7x10^{Scale} Pa <= Pressure <= 3276.7x10^{Scale} Pa corresponding to a *ScaledValue* in the range 0x8001 to 0x7fff.

A *ScaledValue* of 0x8000 indicates that the pressure measurement is invalid.

ScaledValue is updated continuously as new measurements are made.

4.5.2.1.2.2 *MinScaledValue* Attribute

The *MinScaledValue* attribute indicates the minimum value of *ScaledValue* that can be measured. A value of 0x8000 means this attribute is not defined

4.5.2.1.2.3 *MaxScaledValue* Attribute

The *MaxScaledValue* attribute indicates the maximum value of *ScaledValue* that can be measured. A value of 0x8000 means this attribute is not defined.

MaxScaledValue shall be greater than *MinScaledValue.*

MinScaledValue and *MaxScaledValue* define the range of the sensor.

4.5.2.1.2.4 *ScaledTolerance* Attribute

The *ScaledTolerance* attribute indicates the magnitude of the possible error that is associated with *ScaledValue*. The true value is located in the range

(*ScaledValue – ScaledTolerance*) to (*ScaledValue* + *ScaledTolerance*).

4.5.2.1.2.5 *Scale* Attribute

The *Scale* attribute indicates the base 10 exponent used to obtain *ScaledValue* (see 4.5.2.1.2.1).

4.5.2.2 Commands

No cluster specific commands are generated or received by the server cluster.

4.5.2.3 Attribute Reporting

This cluster shall support attribute reporting using the Report Attributes command and according to the minimum and maximum reporting interval and reportable change settings described in the ZCL Foundation specification (see 2.4.7). The following attributes shall be reportable:

MeasuredValue[52]

If the Extended Pressure Measurement Information attribute set is implemented, it is recommended that the following attributes are also reportable:

ScaledValue

ScaledTolerance

4.5.3 Client

The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

4.6 Flow Measurement

4.6.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The server cluster provides an interface to flow measurement functionality, including configuration and provision of notifications of flow measurements.

4.6.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added
2	CCB 2241 2370

4.6.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	FLW	Type 2 (server to client)

4.6.1.3 Cluster Identifiers

Identifier	Name
0x0404	Flow Measurement

[52] CCB 2241 Tolerance is not required to be reportable if supported

4.6.2 Server

4.6.2.1 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets for are listed in Table 4-17.

Table 4-17. Flow Measurement Attribute Sets

Attribute Set Identifier	Description
0x000	Flow Measurement Information

4.6.2.1.1 Flow Measurement Information Attribute Set

The Flow Measurement Information attribute set contains the attributes summarized in Table 4-18.

Table 4-18. Flow Measurement Information Attribute Set

Id	Name	Type	Range	Acc	Def	MO
0x0000	*MeasuredValue*	uint16	*MinMeasuredValue – MaxMeasuredValue*	RP	0xffff[53]	M
0x0001	*MinMeasuredValue*	uint16	0x0000 – 0xfffd	R	0xffff	M
0x0002	*MaxMeasuredValue*	uint16	0x0001 – 0xfffe	R	0xffff	M
0x0003	*Tolerance*	uint16	0x0000 – 0x0800	R[54]		O

4.6.2.1.1.1 *MeasuredValue* Attribute

MeasuredValue represents the flow in m^3/h as follows:

MeasuredValue = 10 x Flow

Where 0 m^3/h <= Flow <= 6,553.4 m^3/h, corresponding to a *MeasuredValue* in the range 0 to 0xfffe.

The maximum resolution this format allows is 0.1 m^3/h.

MinMeasuredValue and *MaxMeasuredValue* define the range of the sensor.

A *MeasuredValue* of 0xffff indicates that the pressure measurement is unknown, otherwise the range SHALL be as described in 4.1.3.

MeasuredValue is updated continuously as new measurements are made.

4.6.2.1.1.2 *MinMeasuredValue* Attribute

The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that can be measured. A value of 0xffff means this attribute is not defined. See 4.1.3 for more details.

[53] CCB 2370 better define default values
[54] CCB 2241 Tolerance is not required to be reportable if supported

4.6.2.1.1.3 *MaxMeasuredValue* Attribute

The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that can be measured. A value of 0xffff means this attribute is not defined. See 4.1.3 for more details.

4.6.2.1.1.4 *Tolerance* Attribute

See 4.1.3.

4.6.2.2 Commands

No cluster specific commands are generated or received by the server cluster.

4.6.2.3 Attribute Reporting

This cluster shall support attribute reporting using the Report Attributes command and according to the minimum and maximum reporting interval and reportable change settings described in the ZCL Foundation specification (see 2.4.7). The following attributes shall be reported:

MeasuredValue[55]

4.6.3 Client

The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

4.7 Water Content[56] Measurement

4.7.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This is a base cluster. The server cluster provides an interface to water content measurement functionality. The measurement is reportable and may be configured for reporting. Water content measurements include, but are not limited to, leaf wetness, relative humidity, and soil moisture.

4.7.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added
2	CCB 2241

4.7.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	RH	Type 2 (server to client)

[55] CCB 2241 Tolerance is not required to be reportable if supported
[56] Formerly was Relative Humidity, now a generic base cluster with multiple Cluster Ids

4.7.1.3 Cluster Identifiers

Identifier	Name	Description
0x0405	Relative Humidity	Percentage of water in the air
0x0407	Leaf Wetness	Percentage of water on the leaves of plants
0x0408	Soil Moisture	Percentage of water in the soil

4.7.2 Server

4.7.2.1 Attributes

Table 4-19. Attributes of the Water Content cluster

Id	Name	Type	Range	Acc	Def	MO
0x0000	*MeasuredValue*	uint16	*MinMeasuredValue – MaxMeasuredValue*	RP	0xffff	M
0x0001	*MinMeasuredValue*	uint16	0x0000 – 0x270f	R	0xffff	M
0x0002	*MaxMeasuredValue*	uint16	0x0001 – 0x2710	R	0xffff	M
0x0003	*Tolerance*	uint16	0x0000 – 0x0800	R[57]		O

4.7.2.1.1 *MeasuredValue* Attribute

MeasuredValue represents the water content in % as follows:

MeasuredValue = 100 x water content

Where 0% <= water content <= 100%, corresponding to a *MeasuredValue* in the range 0 to 0x2710.

The maximum resolution this format allows is 0.01%.

MinMeasuredValue and *MaxMeasuredValue* define the range of the sensor.

A *MeasuredValue* of 0xffff indicates that the measurement is unknown, otherwise the range SHALL be as described in 4.1.3.

MeasuredValue is updated continuously as new measurements are made.

4.7.2.1.2 *MinMeasuredValue* Attribute

The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that can be measured. A value of 0xffff means this attribute is not defined. See 4.1.3 for more details.

4.7.2.1.3 *MaxMeasuredValue* Attribute

The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that can be measured. A value of 0xffff means this attribute is not defined. See 4.1.3 for more details.

[57] CCB 2241 Tolerance is not required to be reportable if supported

4.7.2.1.4 *Tolerance* Attribute

See 4.1.3.

4.7.2.2 Commands

No cluster specific commands are received or generated by the server cluster.

4.7.3 Client

The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

4.8 Occupancy Sensing

4.8.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The server cluster provides an interface to occupancy sensing functionality, including configuration and provision of notifications of occupancy status.

4.8.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added
2	Physical Contact Occupancy feature with mandatory *OccupancySensorTypeBitmap*

4.8.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	OCC	Type 2 (server to client)

4.8.1.3 Cluster Identifiers

Identifier	Name
0x0406	Occupancy Sensing

4.8.2 Server

4.8.2.1 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 4-20.

Table 4-20. Occupancy Sensor Attribute Sets

Attribute Set Identifier	Description
0x000	Occupancy sensor information
0x001	PIR configuration
0x002	Ultrasonic configuration
0x003	Physical contact configuration[58]

4.8.2.1.1 Occupancy Sensor Information Set

The occupancy sensor information attribute set contains the attributes summarized in Table 4-21.

Table 4-21. Occupancy Sensor Information Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0000	*Occupancy*	map8	0b0000 000x	RP	-	M
0x0001	*OccupancySensorType*	enum8		R	MS	M
0x0002	*OccupancySensorTypeBitmap*[59]	map8	0000 0xxx	R	-	M

4.8.2.1.1.1 *Occupancy* Attribute

The *Occupancy* attribute is a bitmap.

Bit 0 specifies the sensed occupancy as follows: 1 = occupied, 0 = unoccupied.

All other bits are reserved.

4.8.2.1.1.2 *OccupancySensorType* Attribute

The *OccupancySensorType* attribute specifies the type of the occupancy sensor. This attribute shall be set to one of the non-reserved values listed in Table 4-22.

Table 4-22. Values of the *OccupancySensorType* Attribute

Attribute Value	Description
0x00	PIR
0x01	Ultrasonic
0x02	PIR and ultrasonic
0x03	Physical contact[60]

4.8.2.1.1.3 OccupancySensorTypeBitmap Attribute

[58] NFR Physical Contact Occupancy Sensor
[59] NFR Physical Contact Occupancy Sensor
[60] NFR Physical Contact Occupancy Sensor

The *OccupancySensorTypeBitmap* attribute specifies the types of the occupancy sensor, as listed below; a ‘1’ in each bit position indicates this type is implemented.

Table 4-23. The *OccupancySensorTypeBitmap* Attribute

Bit	Description
Bit0	PIR
Bit1	Ultrasonic
Bit2	Physical contact

The value of the *OccupancySensorTypeBitmap* attribute and the *OccupancySensorType* attribute SHALL be aligned as defined below.

Table 4-24. Mapping between *OccupancySensorType and OccupancySensorTypeBitmap* Attributes

Description	*OccupancySensorType* attribute	*OccupancySensorTypeBitmap* attribute
PIR	0x00	0000 0001
Ultrasonic	0x01	0000 0010
PIR and ultrasonic	0x02	0000 0011
Physical contact and PIR	0x00	0000 0101
Physical contact and ultrasonic	0x01	0000 0110
Physical contact and PIR and ultrasonic	0x02	0000 0111

4.8.2.1.2 PIR Configuration Set

The PIR sensor configuration attribute set contains the attributes summarized in Table 4-25.

Table 4-25. Attributes of the PIR Configuration Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0010	*PIROccupiedToUnoccupiedDelay*	uint16	0x00 – 0xfffe	RW	0x00	O
0x0011	*PIRUnoccupiedToOccupiedDelay*	uint16	0x00 – 0xfffe	RW	0x00	O
0x0012	*PIRUnoccupiedToOccupiedThreshold*	uint8	0x01 – 0xfe	RW	0x01	O

4.8.2.1.2.1 *PIROccupiedToUnoccupiedDelay* Attribute

The *PIROccupiedToUnoccupiedDelay* attribute is 16 bits in length and specifies the time delay, in seconds, before the PIR sensor changes to its unoccupied state after the last detection of movement in the sensed area.

4.8.2.1.2.2 *PIRUnoccupiedToOccupiedDelay* Attribute

The *PIRUnoccupiedToOccupiedDelay* attribute is 16 bits in length and specifies the time delay, in seconds, before the PIR sensor changes to its occupied state after the detection of movement in the sensed area. This attribute is mandatory if the *PIRUnoccupiedToOccupiedThreshold* attribute is implemented.

4.8.2.1.2.3 *PIRUnoccupiedToOccupiedThreshold* Attribute

The *PIRUnoccupiedToOccupiedThreshold* attribute is 8 bits in length and specifies the number of movement detection events that must occur in the period *PIRUnoccupiedToOccupiedDelay*, before the PIR sensor changes to its occupied state. This attribute is mandatory if the *PIRUnoccupiedToOccupiedDelay* attribute is implemented.

4.8.2.1.3 Ultrasonic Configuration Set

The ultrasonic sensor configuration attribute set contains the attributes summarized in Table 4-26.

Table 4-26. Attributes of the Ultrasonic Configuration Attribute Set

Id	Name	Type	Range	Acc	Def	MO
0x0020	*UltrasonicOccupiedToUnoccupiedDelay*	uint16	0x0000 – 0xfffe	RW	0x00	O
0x0021	*UltrasonicUnoccupiedToOccupiedDelay*	uint16	0x0000 – 0xfffe	RW	0x00	O
0x0022	*UltrasonicUnoccupiedToOccupiedThreshold*	uint8	0x01 – 0xfe	RW	0x01	O

4.8.2.1.3.1 *UltrasonicOccupiedToUnoccupiedDelay* Attribute

The *UltrasonicOccupiedToUnoccupiedDelay* attribute is 16 bits in length and specifies the time delay, in seconds, before the Ultrasonic sensor changes to its unoccupied state after the last detection of movement in the sensed area.

4.8.2.1.3.2 *UltrasonicUnoccupiedToOccupiedDelay* Attribute

The *UltrasonicUnoccupiedToOccupiedDelay* attribute is 16 bits in length and specifies the time delay, in seconds, before the Ultrasonic sensor changes to its occupied state after the detection of movement in the sensed area. This attribute is mandatory if the *UltrasonicUnoccupiedToOccupiedThreshold* attribute is implemented.

4.8.2.1.3.3 *UltrasonicUnoccupiedToOccupiedThreshold* Attribute

The *UltrasonicUnoccupiedToOccupiedThreshold* attribute is 8 bits in length and specifies the number of movement detection events that must occur in the period *UltrasonicUnoccupiedToOccupiedDelay*, before the Ultrasonic sensor changes to its occupied state. This attribute is mandatory if the *UltrasonicUnoccupiedToOccupiedDelay* attribute is implemented.

4.8.2.1.4 Physical Contact Configuration Set[61]

The physical contact configuration attribute set contains the attributes summarized below.

Table 4-27. Attributes of the Physical Contact Configuration Attribute Set

Id	Name	Type	Range	Acc	Def	MO

[61] NFR Physical Contact Occupancy Sensor

0x0030	*PhysicalContactOccupiedToUnoccupiedDelay*	uint16	0x0000 to 0xfffe	RW	0x0000	O
0x0031	*PhysicalContactUnoccupiedToOccupiedDelay*	uint16	0x0000 to 0xfffe	RW	0x0000	O
0x0032	*PhysicalContactUnoccupiedToOccupiedThreshold*	uint8	0x01 to 0xfe	RW	0x01	O

4.8.2.1.4.1 *PhysicalContactOccupiedToUnoccupiedDelay* Attribute

The *PhysicalContactOccupiedToUnoccupiedDelay* attribute is 16 bits in length and specifies the time delay, in seconds, before the physical contact occupancy sensor changes to its unoccupied state after detecting the unoccupied event. The value of 0xffff indicates the sensor does not report occupied to unoccupied transition.

4.8.2.1.4.2 *PhysicalContactUnoccupiedToOccupiedDelay* Attribute

The *PhysicalContactUnoccupiedToOccupiedDelay* attribute is 16 bits in length and specifies the time delay, in seconds, before the physical contact sensor changes to its occupied state after the detection of the occupied event.

The value of 0xffff indicates the sensor does not report unoccupied to occupied transition.

4.8.2.1.4.3 *PhysicalContactUnoccupiedToOccupiedThreshold* Attribute

The *PhysicalContactUnoccupiedToOccupiedThreshold* attribute is 8 bits in length and specifies the number of movement detection events that must occur in the period *PhysicalContactUnoccupiedToOccupiedDelay*, before the PIR sensor changes to its occupied state. This attribute is mandatory if the *PhysicalContactUnoccupiedToOccupiedDelay* attribute is implemented.

4.8.2.2 Commands

No cluster specific commands are received by the server cluster. No cluster specific commands are generated by the server cluster.

4.8.3 Client

The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

4.9 Electrical Measurement

4.9.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides a mechanism for querying data about the electrical properties as measured by the device. This cluster may be implemented on any device type and be implemented on a per-endpoint basis. For example, a power strip device could represent each outlet on a different endpoint and report electrical information for each individual outlet. The only caveat is that if you implement an attribute that has an associated multiplier and divisor, then you must implement the associated multiplier and divisor attributes. For example if you implement DCVoltage, you must also implement DCVoltageMultiplier and DCVoltageDivisor.

If you are interested in reading information about the power supply or battery level on the device, please see the Power Configuration cluster.

4.9.1.1 Revision History

Rev	Description

1	mandatory global *ClusterRevision* attribute added
2	CCB 2236

4.9.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	EMR	Type 1 (client to server)

4.9.1.3 Cluster Identifiers

Identifier	Name
0x0b04	Electrical Measurement

4.9.1.4 Formatting

Most measurement values have an associated multiplier and divisor attribute. Multiplier attributes provide a value to be multiplied against a raw or uncompensated measurement value. Divisor attributes provide a value to divide the results of applying a multiplier attribute against a raw or uncompensated measurement value. If a multiplier or divisor attribute is present, its corresponding divisor or multiplier attribute shall be implemented as well.

4.9.2 Server

4.9.2.1 Dependencies

For the alarm functionality in this cluster to be operational, any endpoint that implements the Electrical Measurement server cluster shall also implement the Alarms server cluster.

4.9.2.2 Attributes

The server side of this cluster contains certain attributes associated with the electrical properties and configuration, as shown in Table 4-28.

Table 4-28. Attributes of the Electrical Measurement Cluster

Attribute Set Identifier	Description
0x00	Basic Information
0x01	DC Measurement
0x02	DC Formatting
0x03	AC (Non-phase Specific) Measurements
0x04	AC (Non-phase Specific) Formatting
0x05	AC (Single Phase or Phase A) Measurements
0x06	AC Formatting

Attribute Set Identifier	Description
0x07	DC Manufacturer Threshold Alarms
0x08	AC Manufacturer Threshold Alarms
0x09	AC Phase B Measurements
0x0a	AC Phase C Measurements

4.9.2.2.1 Basic Information

Table 4-29. Electrical Measurement Cluster Basic Information

Id	Name	Type	Range	Acc	Def	MO
0x0000	*MeasurementType*	map32	0x00000000 – 0xffffFFFF	R	0x00000000	M

4.9.2.2.1.1 *MeasurementType*

This attribute indicates a device's measurement capabilities. This will be indicated by setting the desire measurement bits to 1, as mentioned in Table 4-30 and DC Measurement

Table 4-31. This attribute will be used client devices to determine what all attribute is supported by the meter. Unused bits should be set to zero.

Table 4-30. *MeasurementType* Attribute

Bit	Flag Name / Description
0	Active measurement (AC)
1	Reactive measurement (AC)
2	Apparent measurement (AC)
3	Phase A measurement
4	Phase B measurement
5	Phase C measurement
6	DC measurement
7	Harmonics measurement
8	Power quality measurement

4.9.2.2.2 DC Measurement

Table 4-31. DC Measurement Attributes

Id	Name	Type	Range	Acc	Default	M/O
0x0100	*DCVoltage*	int16	-32767 – 32767	RP	0x8000	O
0x0101	*DCVoltageMin*	int16	-32767 – 32767	R	0x8000	O
0x0102	*DCVoltageMax*	int16	-32767 – 32767	R	0x8000	O
0x0103	*DCCurrent*	int16	-32767 – 32767	RP	0x8000	O
0x0104	*DCCurrentMin*	int16	-32767 – 32767	R	0x8000	O
0x0105	*DCCurrentMax*	int16	-32767 – 32767	R	0x8000	O
0x0106	*DCPower*	int16	-32767 – 32767	RP	0x8000	O
0x0107	*DCPowerMin*	int16	-32767 – 32767	R	0x8000	O
0x0108	*DCPowerMax*	int16	-32767 – 32767	R	0x8000	O
0x0109 – 0x01FF	Reserved					

4.9.2.2.2.1 *DCVoltage*

The DCVoltage attribute represents the most recent DC voltage reading in Volts (V). If the voltage cannot be measured, a value of 0x8000 is returned.

4.9.2.2.2.2 *DCVoltageMin*

The DCVoltageMin attribute represents the lowest DC voltage value measured in Volts (V). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.2.3 *DCVoltageMax*

The DCVoltageMax attribute represents the highest DC voltage value measured in Volts (V). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.2.4 *DCCurrent*

The DCCurrent attribute represents the most recent DC current reading in Amps (A). If the current cannot be measured, a value of 0x8000 is returned.

4.9.2.2.2.5 *DCCurrentMin*

The DCCurrentMin attribute represents the lowest DC current value measured in Amps (A). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.2.6 *DCCurrentMax*

The DCCurrentMax attribute represents the highest DC current value measured in Amps (A). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.2.7 DCPower

The DCPower attribute represents the most recent DC power reading in Watts (W). If the power cannot be measured, a value of 0x8000 is returned.

4.9.2.2.2.8 DCPowerMin

The DCPowerMin attribute represents the lowest DC power value measured in Watts (W). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.2.9 DCPowerMax

The DCPowerMax attribute represents the highest DC power value measured in Watts (W). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.3 DC Formatting

Table 4-32. DC Formatting Attributes

Id	Name	Type	Range	Access	Default	M/O
0x0200	*DCVoltageMultiplier*	uint16	0x0001 – 0xffff	RP[62]	0x0001	O
0x0201	*DCVoltageDivisor*	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0202	*DCCurrentMultiplier*	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0203	*DCCurrentDivisor*	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0204	*DCPowerMultiplier*	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0205	*DCPowerDivisor*	uint16	0x0001 – 0xffff	RP	0x0001	O

4.9.2.2.3.1 DCVoltageMultiplier

The *DCVoltageMultiplier* provides a value to be multiplied against the *DCVoltage*, *DCVoltageMin*, and *DCVoltageMax* attributes. This attribute must be used in conjunction with the *DCVoltageDivisor* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.3.2 DCVoltageDivisor

The *DCVoltageDivisor* provides a value to be divided against the *DCVoltage*, *DCVoltageMin*, and *DCVoltageMax* attributes. This attribute must be used in conjunction with the *DCVoltageMultiplier* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.3.3 DCCurrentMultiplier

The *DCCurrentMultiplier* provides a value to be multiplied against the *DCCurrent*, *DCCurrentMin*, and *DCCurrentMax* attributes. This attribute must be used in conjunction with the *DCCurrentDivisor* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.3.4 DCCurrentDivisor

[62] CCB 2236 for all multipliers and divisors that may change, though less frequently than the values

The *DCCurrentDivisor* provides a value to be divided against the *DCCurrent*, *DCCurrentMin*, and *DCCurrentMax* attributes. This attribute must be used in conjunction with the *DCCurrentMultiplier* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.3.5 *DCPowerMultiplier*

The *DCPowerMultiplier* provides a value to be multiplied against the *DCPower*, *DCPowerMin*, and *DCPowerMax* attributes. This attribute must be used in conjunction with the *DCPowerDivisor* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.3.6 *DCPowerDivisor*

The *DCPowerDivisor* provides a value to be divided against the *DCPower*, *DCPowerMin*, and *DCPowerMax* attributes. This attribute must be used in conjunction with the *DCPowerMultiplier* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.4 AC (Non-phase Specific) Measurements

Table 4-33. AC (Non-phase Specific) Measurement Attributes

Id	Name	Type	Range	Acc	Default	M/O
0x0300	*ACFrequency*	uint16	0x0000 – 0xffff	RP[63]	0xffff	O
0x0301	*ACFrequencyMin*	uint16	0x0000 – 0xffff	R	0xffff	O
0x0302	*ACFrequencyMax*	uint16	0x0000 – 0xffff	R	0xffff	O
0x0303	*NeutralCurrent*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0304	*TotalActivePower*	int32	-8,388,607–8,388,607	RP	-	O
0x0305	*TotalReactivePower*	int32	-8,388,607–8,388,607	RP	-	O
0x0306	*TotalApparentPower*	uint32	0x000000–0xffffFF	RP	-	O
0x0307	*Measured1stHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x0308	*Measured3rdHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x0309	*Measured5thHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x030a	*Measured7thHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x030b	*Measured9thHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x030c	*Measured11thHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x030d	*MeasuredPhase1stHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x030e	*MeasuredPhase3rdHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O

[63] CCB 2236 for all changing values

Id	Name	Type	Range	Acc	Default	M/O
0x030f	*MeasuredPhase5thHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x0310	*MeasuredPhase7thHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x0311	*MeasuredPhase9thHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x0312	*MeasuredPhase11thHarmonicCurrent*	int16	-32768 – 32767	RP	0x8000	O

4.9.2.2.4.1 *ACFrequency*

The *ACFrequency* attribute represents the most recent AC Frequency reading in Hertz (Hz). If the frequency cannot be measured, a value of 0xffff is returned.

4.9.2.2.4.2 *ACFrequencyMin*

The *ACFrequencyMin* attribute represents the lowest AC Frequency value measured in Hertz (Hz). After resetting, this attribute will return a value of 0xffff until a measurement is made.

4.9.2.2.4.3 *ACFrequencyMax*

The *ACFrequencyMax* attribute represents the highest AC Frequency value measured in Hertz (Hz). After resetting, this attribute will return a value of 0xffff until a measurement is made.

4.9.2.2.4.4 *NeutralCurrent*

The *NeutralCurrent* attribute represents the AC neutral (Line-Out) current value at the moment in time the attribute is read, in Amps (A). If the instantaneous current cannot be measured, a value of 0xffff is returned.

4.9.2.2.4.5 *TotalActivePower*

Active power represents the current demand of active power delivered or received at the premises, in kW. Positive values indicate power delivered to the premises where negative values indicate power received from the premises. In case if device is capable of measuring multi elements or phases then this will be net active power value.

4.9.2.2.4.6 *TotalReactivePower*

Reactive power represents the current demand of reactive power delivered or received at the premises, in kVAr. Positive values indicate power delivered to the premises where negative values indicate power received from the premises. In case if device is capable of measuring multi elements or phases then this will be net reactive power value.

4.9.2.2.4.7 *TotalApparentPower*

Represents the current demand of apparent power, in kVA. In case if device is capable of measuring multi elements or phases then this will be net apparent power value.

4.9.2.2.4.8 *MeasuredNthHarmonicCurrent* Attributes

The *Measured1stHarmonicCurrent* through *MeasuredNthHarmonicCurrent* attributes represent the most recent Nth harmonic current reading in an AC frequency. The unit for this measurement is 10 ^ *NthHarmonicCurrentMultiplier* amperes. If *NthHarmonicCurrentMultiplier* is not implemented the unit is in amperes. If the Nth harmonic current cannot be measured a value of 0x8000 is returned. A positive value indicates the measured Nth harmonic current is positive, and a negative value indicates that the measured Nth harmonic current is negative.

4.9.2.2.4.9 *MeasuredPhaseNthHarmonicCurrent* Attributes

The *MeasuredPhase1stHarmonicCurrent* through *MeasuredPhaseNthHarmonicCurrent* attributes represent the most recent phase of the Nth harmonic current reading in an AC frequency. The unit for this measurement is 10 ^ *PhaseNthHarmonicCurrentMultiplier* degree. If *PhaseNthHarmonicCurrentMultiplier* is not implemented the unit is in degree. If the phase of the Nth harmonic current cannot be measured a value of 0x8000 is returned. A positive value indicates the measured phase of the Nth harmonic current is prehurry, and a negative value indicates that the measured phase of the Nth harmonic current is lagging.

4.9.2.2.5 AC (Non-phase Specific) Formatting

Table 4-34. AC (Non-phase Specific) Formatting Attributes

Id	Name	Type	Range	Acc	Default	M/O
0x0400	*ACFrequencyMultiplier*	uint16	0x0001 – 0xffff	RP[64]	0x0001	O
0x0401	*ACFrequencyDivisor*	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0402	*PowerMultiplier*	uint32	0x000000 – 0xffffFF	RP	0x000001	O
0x0403	*PowerDivisor*	uint32	0x00000 – 0xffffFF	RP	0x000001	O
0x0404	*HarmonicCurrentMultiplier*	int8	-127 – 127	RP	0x00	O
0x0405	*PhaseHarmonicCurrentMultiplier*	int8	-127 – 127	RP	0x00	O

4.9.2.2.5.1 *ACFrequencyMultiplier*

Provides a value to be multiplied against the *ACFrequency* attribute. This attribute must be used in conjunction with the *ACFrequencyDivisor* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.5.2 *ACFrequencyDivisor*

Provides a value to be divided against the *ACFrequency* attribute. This attribute must be used in conjunction with the *ACFrequencyMultiplier* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.5.3 *PowerMultiplier*

Provides a value to be multiplied against a raw or uncompensated sensor count of power being measured by the metering device. If present, this attribute must be applied against all power/demand values to derive the delivered and received values expressed in the specified units. This attribute must be used in conjunction with the *PowerDivisor* attribute.

4.9.2.2.5.4 *PowerDivisor*

Provides a value to divide against the results of applying the *Multiplier* attribute against a raw or uncompensated sensor count of power being measured by the metering device. If present, this attribute must be applied against all demand/power values to derive the delivered and received values expressed in the specified units. This attribute must be used in conjunction with the *PowerMultiplier* attribute.

4.9.2.2.5.5 *HarmonicCurrentMultiplier*

Represents the unit value for the *MeasuredNthHarmonicCurrent* attribute in the format *MeasuredNthHarmonicCurrent* * 10 ^ *HarmonicCurrentMultiplier* amperes.

[64] CCB 2236 for all multipliers and divisors that may change, though less frequently than the values

4.9.2.2.5.6 *PhaseHarmonicCurrentMultiplier*

Represents the unit value for the *MeasuredPhaseNthHarmonicCurrent* attribute in the format *MeasuredPhaseNthHarmonicCurrent* * 10 ^ *PhaseHarmonicCurrentMultiplier* degrees.

4.9.2.2.6 AC (Single Phase or Phase A) Measurements

Table 4-35. AC (Single Phase or Phase A) Measurement Attributes

Id	Name	Type	Range	Acc	Default	M/O
0x0501	*LineCurrent*	uint16	0x0000 – 0xffff	RP[65]	0xffff	O
0x0502	*ActiveCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x0503	*ReactiveCurrent*	int16	-32768 – 32767	RP	0x8000	O
0x0505	*RMSVoltage*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0506	*RMSVoltageMin*	uint16	0x0000 – 0xffff	R	0xffff	O
0x0507	*RMSVoltageMax*	uint16	0x0000 – 0xffff	R	0xffff	O
0x0508	*RMSCurrent*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0509	*RMSCurrentMin*	uint16	0x0000 – 0xffff	R	0xffff	O
0x050a	*RMSCurrentMax*	uint16	0x0000 – 0xffff	R	0xffff	O
0x050b	*ActivePower*	int16	-32768 – 32767	RP	0x8000	O
0x050c	*ActivePowerMin*	int16	-32768 – 32767	R	0x8000	O
0x050d	*ActivePowerMax*	int16	-32768 – 32767	R	0x8000	O
0x050e	*ReactivePower*	int16	-32768 – 32767	RP	0x8000	O
0x050f	*ApparentPower*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0510	*PowerFactor*	int8	-100 to +100	R	0x00	O
0x0511	*AverageRMSVoltageMeasurementPeriod*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0512	*AverageRMSOverVoltageCounter*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0513	*AverageRMSUnderVoltageCounter*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0514	*RMSExtremeOverVoltagePeriod*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0515	*RMSExtremeUnderVoltagePeriod*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0516	*RMSVoltageSagPeriod*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0517	*RMSVoltageSwellPeriod*	uint16	0x0000 – 0xffff	RW	0x0000	O

4.9.2.2.6.1 *LineCurrent*

[65] CCB 2236 for all changing values

Represents the single phase or Phase A, AC line current (Square root of active and reactive current) value at the moment in time the attribute is read, in Amps (A). If the instantaneous current cannot be measured, a value of 0x8000 is returned.

4.9.2.2.6.2 *ActiveCurrent*

Represents the single phase or Phase A, AC active/resistive current value at the moment in time the attribute is read, in Amps (A). Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.6.3 *ReactiveCurrent*

Represents the single phase or Phase A, AC reactive current value at the moment in time the attribute is read, in Amps (A). Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.6.4 *RMSVoltage*

Represents the most recent RMS voltage reading in Volts (V). If the RMS voltage cannot be measured, a value of 0xffff is returned.

4.9.2.2.6.5 *RMSVoltageMin*

Represents the lowest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of 0xffff until a measurement is made.

4.9.2.2.6.6 *RMSVoltageMax*

Represents the highest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of 0xffff until a measurement is made.

4.9.2.2.6.7 *RMSCurrent*

Represents the most recent RMS current reading in Amps (A). If the power cannot be measured, a value of 0xffff is returned.

4.9.2.2.6.8 *RMSCurrentMin*

Represents the lowest RMS current value measured in Amps (A). After resetting, this attribute will return a value of 0xffff until a measurement is made.

4.9.2.2.6.9 *RMSCurrentMax*

Represents the highest RMS current value measured in Amps (A). After resetting, this attribute will return a value of 0xffff until a measurement is made.

4.9.2.2.6.10 *ActivePower*

Represents the single phase or Phase A, current demand of active power delivered or received at the premises, in Watts (W). Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.6.11 *ActivePowerMin*

Represents the lowest AC power value measured in Watts (W). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.6.12 *ActivePowerMax*

Represents the highest AC power value measured in Watts (W). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.6.13 *ReactivePower*

Represents the single phase or Phase A, current demand of reactive power delivered or received at the premises, in VAr. Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.6.14 *ApparentPower*

Represents the single phase or Phase A, current demand of apparent (Square root of active and reactive power) power, in VA.

4.9.2.2.6.15 *PowerFactor*

Contains the single phase or PhaseA, Power Factor ratio in 1/100ths.

4.9.2.2.6.16 *AverageRMSVoltageMeasurementPeriod*

The Period in seconds that the RMS voltage is averaged over.

4.9.2.2.6.17 *AverageRMSOverVoltageCounter*

The number of times the average RMS voltage, has been above the *AverageRMS OverVoltage* threshold since last reset. This counter may be reset by writing zero to the attribute.

4.9.2.2.6.18 *AverageRMSUnderVoltageCounter*

The number of times the average RMS voltage, has been below the *AverageRMS underVoltage* threshold since last reset. This counter may be reset by writing zero to the attribute.

4.9.2.2.6.19 *RMSExtremeOverVoltagePeriod*

The duration in seconds used to measure an extreme over voltage condition.

4.9.2.2.6.20 *RMSExtremeUnderVoltagePeriod*

The duration in seconds used to measure an extreme under voltage condition.

4.9.2.2.6.21 *RMSVoltageSagPeriod*

The duration in seconds used to measure a voltage sag condition.

4.9.2.2.6.22 *RMSVoltageSwellPeriod*

The duration in seconds used to measure a voltage swell condition.

4.9.2.2.7 AC Formatting

Table 4-36. AC Formatting Attributes

Id	Name	Type	Range	Acc	Default	M/O
0x0600	*ACVoltageMultiplier*	uint16	0x0001 – 0xffff	RP[66]	0x0001	O
0x0601	*ACVoltageDivisor*	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0602	*ACCurrentMultiplier*	uint16	0x0001 – 0xffff	RP	0x0001	O

[66] CCB 2236 for all multipliers and divisors that may change, though less frequently than the values

Id	Name	Type	Range	Acc	Default	M/O
0x0603	*ACCurrentDivisor*	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0604	*ACPowerMultiplier*	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0605	*ACPowerDivisor*	uint16	0x0001 – 0xffff	RP	0x0001	O

4.9.2.2.7.1 *ACVoltageMultiplier*

Provides a value to be multiplied against the *InstantaneousVoltage* and *RMSVoltage* attributes. This attribute must be used in conjunction with the *ACVoltageDivisor* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.7.2 *ACVoltageDivisor*

Provides a value to be divided against the *InstantaneousVoltage* and *RMSVoltage* attributes. This attribute must be used in conjunction with the *ACVoltageMultiplier* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.7.3 *ACCurrentMultiplier*

Provides a value to be multiplied against the *InstantaneousCurrent* and *RMSCurrent* attributes. This attribute must be used in conjunction with the *ACCurrentDivisor* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.7.4 *ACCurrentDivisor*

Provides a value to be divided against the *ACCurrent*, *InstantaneousCurrent* and *RMSCurrent* attributes. This attribute must be used in conjunction with the *ACCurrentMultiplier* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.7.5 *ACPowerMultiplier*

Provides a value to be multiplied against the *InstantaneousPower* and *ActivePower* attributes. This attribute must be used in conjunction with the *ACPowerDivisor* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.7.6 *ACPowerDivisor*

Provides a value to be divided against the *InstantaneousPower* and *ActivePower* attributes. This attribute must be used in conjunction with the *ACPowerMultiplier* attribute. 0x0000 is an invalid value for this attribute.

4.9.2.2.8 DC Manufacturer Threshold Alarms

Table 4-37. DC Manufacturer Threshold Alarms Attributes

Id	Name	Type	Range	Access	Default	M/O
0x0700	*DCOverloadAlarmsMask*	map8	0000 00xx	RW	0000 0000	O
0x0701	*DCVoltageOverload*	int16	-32768 – 32767	R	0xffff	O
0x0702	*DCCurrentOverload*	int16	-32768 – 32767	R	0xffff	O

4.9.2.2.8.1 *DCOverloadAlarmsMask*

Specifies which configurable alarms may be generated, as listed in Figure 4-4. A ‘1’ in each bit position enables the alarm.

Figure 4-4. The DC Overload Alarm Mask

Bit	Description
Bit0	Voltage Overload
Bit1	Current Overload

4.9.2.2.8.2 *DCVoltageOverload*

Specifies the alarm threshold, set by the manufacturer, for the maximum output voltage supported by device. The value is multiplied and divided by the *DCVoltageMultiplier* the *DCVoltageDivisor* respectively.

4.9.2.2.8.3 *DCCurrentOverload*

Specifies the alarm threshold, set by the manufacturer, for the maximum output current supported by device. The value is multiplied and divided by the *DCCurrentMultiplier* and *DCCurrentDivider* respectively.

4.9.2.2.9 AC Manufacturer Threshold Alarms

Table 4-38. AC Manufacturer Threshold Alarms Attributes

Id	Name	Type	Range	Access	Default	MO
0x0800	*ACAlarmsMask*	map16	0000 xxxx	RW	0000 0000	O
0x0801	*ACVoltageOverload*	int16	-32768 – 32767	R	0xffff	O
0x0802	*ACCurrentOverload*	int16	-32768 – 32767	R	0xffff	O
0x0803	*ACActivePowerOverload*	int16	-32768 – 32767	R	0xffff	O
0x0804	*ACReactivePowerOverload*	int16	-32768 – 32767	R	0xffff	O
0x0805	*AverageRMSOverVoltage*	int16	-32768 – 32767	R		O
0x0806	*AverageRMSUnderVoltage*	int16	-32768 – 32767	R		O
0x0807	*RMSExtremeOverVoltage*	int16	-32768 – 32767	RW		O
0x0808	*RMSExtremeUnderVoltage*	int16	-32768 – 32767	RW		O
0x0809	*RMSVoltageSag*	int16	-32768 – 32767	RW		O
0x080a	*RMSVoltageSwell*	int16	-32768 – 32767	RW		O

4.9.2.2.9.1 *ACAlarmsMask*

Specifies which configurable alarms may be generated, as listed in Figure 4-5. A ‘1’ in each bit position enables the alarm.

Figure 4-5. The *ACAlarmsMask* Attribute

Bit	Description
Bit0	Voltage Overload
Bit1	Current Overload
Bit2	Active Power Overload
Bit3	Reactive Power Overload
Bit4	Average RMS Over Voltage
Bit5	Average RMS Under Voltage
Bit6	RMS Extreme Over Voltage
Bit7	RMS Extreme Under Voltage
Bit8	RMS Voltage Sag
Bit9	RMS Voltage Swell

4.9.2.2.9.2 *ACVoltageOverload*

Specifies the alarm threshold, set by the manufacturer, for the maximum output voltage supported by device. The value is multiplied and divided by the A*CVoltageMultiplier* the A*CVoltageDivisor,* respectively. The value is voltage RMS.

4.9.2.2.9.3 *ACCurrentOverload*

Specifies the alarm threshold, set by the manufacturer, for the maximum output current supported by device. The value is multiplied and divided by the A*CCurrentMultiplier* and *ACCurrentDivider,* respectively. The value is current RMS.

4.9.2.2.9.4 *ACActivePowerOverload*

Specifies the alarm threshold, set by the manufacturer, for the maximum output active power supported by device. The value is multiplied and divided by the *ACPowerMultiplier* and *ACPowerDivisor*, respectively.

4.9.2.2.9.5 *ACReactivePowerOverload*

Specifies the alarm threshold, set by the manufacturer, for the maximum output reactive power supported by device. The value is multiplied and divided by the *ACPowerMultiplier* and *ACPowerDivisor*, respectively.

4.9.2.2.9.6 *AverageRMSOverVoltage*

The average RMS voltage above which an over voltage condition is reported. The threshold shall be configurable within the specified operating range of the electricity meter. The value is multiplied and divided by the *ACVoltageMultiplier* and *ACVoltageDivisor*, respectively.

4.9.2.2.9.7 *AverageRMSUnderVoltage*

The average RMS voltage below which an under voltage condition is reported. The threshold shall be configurable within the specified operating range of the electricity meter. The value is multiplied and divided by the *ACVoltageMultiplier* and *ACVoltageDivisor*, respectively.

4.9.2.2.9.8 *RMSExtremeOverVoltage*

The RMS voltage above which an extreme under voltage condition is reported. The threshold shall be configurable within the specified operating range of the electricity meter. The value is multiplied and divided by the *ACVoltageMultiplier* and *ACVoltageDivisor*, respectively.

4.9.2.2.9.9 *RMSExtremeUnderVoltage*

The RMS voltage below which an extreme under voltage condition is reported. The threshold shall be configurable within the specified operating range of the electricity meter. The value is multiplied and divided by the *ACVoltageMultiplier* and *ACVoltageDivisor*, respectively.

4.9.2.2.9.10 *RMSVoltageSag*

The RMS voltage below which a sag condition is reported. The threshold shall be configurable within the specified operating range of the electricity meter. The value is multiplied and divided by the *ACVoltageMultiplier* and *ACVoltageDivisor*, respectively.

4.9.2.2.9.11 *RMSVoltageSwell*

The RMS voltage above which a swell condition is reported. The threshold shall be configurable within the specified operating range of the electricity meter. The value is multiplied and divided by the *ACVoltageMultiplier* and *ACVoltageDivisor*, respectively.

4.9.2.2.10 AC Phase B Measurements

Table 4-39. AC Phase B Measurements Attributes

Id	Name	Type	Range	Acc	Def	M/O
0x0901	*LineCurrentPhB*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0902	*ActiveCurrentPhB*	int16	-32768 – 32767	RP	0x8000	O
0x0903	*ReactiveCurrentPhB*	int16	-32768 – 32767	RP	0x8000	O
0x0905	*RMSVoltagePhB*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0906	*RMSVoltageMinPhB*	uint16	0x0000 – 0xffff	R	0x8000	O
0x0907	*RMSVoltageMaxPhB*	uint16	0x0000 – 0xffff	R	0x8000	O
0x0908	*RMSCurrentPhB*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0909	*RMSCurrentMinPhB*	uint16	0x0000 – 0xffff	R	0xffff	O
0x090a	*RMSCurrentMaxPhB*	uint16	0x0000 – 0xffff	R	0xffff	O
0x090b	*ActivePowerPhB*	int16	-32768 – 32767	RP	0x8000	O
0x090c	*ActivePowerMinPhB*	int16	-32768 – 32767	R	0x8000	O
0x090d	*ActivePowerMaxPhB*	int16	-32768 – 32767	R	0x8000	O

Id	Name	Type	Range	Acc	Def	M/O
0x090e	*ReactivePowerPhB*	int16	-32768 – 32767	RP	0x8000	O
0x090f	*ApparentPowerPhB*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0910	*PowerFactorPhB*	int8	-100 to +100	R	0x00	O
0x0911	*AverageRMSVoltageMeasurementPeriodPhB*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0912	*AverageRMSOverVoltageCounterPhB*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0913	*AverageRMSUnderVoltageCounterPhB*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0914	*RMSExtremeOverVoltagePeriodPhB*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0915	*RMSExtremeUnderVoltagePeriodPhB*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0916	*RMSVoltageSagPeriodPhB*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0917	*RMSVoltageSwellPeriodPhB*	uint16	0x0000 – 0xffff	RW	0x0000	O

4.9.2.2.10.1 *LineCurrentPhB*

Represents the Phase B, AC line current (Square root sum of active and reactive currents) value at the moment in time the attribute is read, in Amps (A). If the instantaneous current cannot be measured, a value of 0x8000 is returned.

4.9.2.2.10.2 *ActiveCurrentPhB*

Represents the Phase B, AC active/resistive current value at the moment in time the attribute is read, in Amps (A). Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.10.3 *ReactiveCurrentPhB*

Represents the Phase B, AC reactive current value at the moment in time the attribute is read, in Amps (A). Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.10.4 *RMSVoltagePhB*

Represents the most recent RMS voltage reading in Volts (V). If the RMS voltage cannot be measured, a value of 0xffff is returned.

4.9.2.2.10.5 *RMSVoltageMinPhB*

Represents the lowest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of 0xffff until a measurement is made.

4.9.2.2.10.6 *RMSVoltageMaxPhB*

Represents the highest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of 0xffff until a measurement is made.

4.9.2.2.10.7 *RMSCurrentPhB*

Represents the most recent RMS current reading in Amps (A). If the power cannot be measured, a value of 0xffff is returned.

4.9.2.2.10.8 *RMSCurrentMinPhB*

Represents the lowest RMS current value measured in Amps (A). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.10.9 *RMSCurrentMaxPhB*

Represents the highest RMS current value measured in Amps (A). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.10.10 *ActivePowerPhB*

Represents the Phase B, current demand of active power delivered or received at the premises, in Watts (W). Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.10.11 *ActivePowerMinPhB*

Represents the lowest AC power value measured in Watts (W). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.10.12 *ActivePowerMaxPhB*

Represents the highest AC power value measured in Watts (W). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.10.13 *ReactivePowerPhB*

Represents the Phase B, current demand of reactive power delivered or received at the premises, in VAr. Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.10.14 *ApparentPowerPhB*

Represents the Phase B, current demand of apparent (Square root of active and reactive power) power, in VA.

4.9.2.2.10.15 *PowerFactorPhB*

Contains the PhaseB, Power Factor ratio in 1/100ths.

4.9.2.2.10.16 *AverageRMSVoltageMeasurementPeriodPhB*

The Period in seconds that the RMS voltage is averaged over.

4.9.2.2.10.17 *AverageRMSOverVoltageCounterPhB*

The number of times the average RMS voltage, has been above the *AverageRMS OverVoltage* threshold since last reset. This counter may be reset by writing zero to the attribute.

4.9.2.2.10.18 *AverageRMSUnderVoltageCounterPhB*

The number of times the average RMS voltage, has been below the *AverageRMS underVoltage* threshold since last reset. This counter may be reset by writing zero to the attribute.

4.9.2.2.10.19 *RMSExtremeOverVoltagePeriodPhB*

The duration in seconds used to measure an extreme over voltage condition.

4.9.2.2.10.20 *RMSExtremeUnderVoltagePeriodPhB*

The duration in seconds used to measure an extreme under voltage condition.

4.9.2.2.10.21 *RMSVoltageSagPeriodPhB*

The duration in seconds used to measure a voltage sag condition.

4.9.2.2.10.22 *RMSVoltageSwellPeriodPhB*

The duration in seconds used to measure a voltage swell condition.

4.9.2.2.11 AC Phase C Measurements

Table 4-40. AC Phase C Measurements Attributes

Id	Name	Type	Range	Acc	Def	M/O
0x0a01	*LineCurrentPhC*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0a02	*ActiveCurrentPhC*	int16	-32768 – 32767	RP	0x8000	O
0x0a03	*ReactiveCurrentPhC*	int16	-32768 – 32767	RP	0x8000	O
0x0a05	*RMSVoltagePhC*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0a06	*RMSVoltageMinPhC*	uint16	0x0000 – 0xffff	R	0x8000	O
0x0a07	*RMSVoltageMaxPhC*	uint16	0x0000 – 0xffff	R	0x8000	O
0x0a08	*RMSCurrentPhC*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0a09	*RMSCurrentMinPhC*	uint16	0x0000 – 0xffff	R	0xffff	O
0x0a0a	*RMSCurrentMaxPhC*	uint16	0x0000 – 0xffff	R	0xffff	O
0x0a0b	*ActivePowerPhC*	int16	-32768 – 32767	RP	0x8000	O
0x0a0c	*ActivePowerMinPhC*	int16	-32768 – 32767	R	0x8000	O
0x0a0d	*ActivePowerMaxPhC*	int16	-32768 – 32767	R	0x8000	O
0x0a0e	*ReactivePowerPhC*	int16	-32768 – 32767	RP	0x8000	O
0x0a0f	*ApparentPowerPhC*	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0a10	*PowerFactorPhC*	int8	-100 to +100	R	0x00	O
0x0a11	*AverageRMSVoltageMeasurementPeriodPhC*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0a12	*AverageRMSOverVoltageCounterPhC*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0a13	*AverageRMSUnderVoltageCounterPhC*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0a14	*RMSExtremeOverVoltagePeriodPhC*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0a15	*RMSExtremeUnderVoltagePeriodPhC*	uint16	0x0000 – 0xffff	RW	0x0000	O

Id	Name	Type	Range	Acc	Def	M/O
0x0a16	*RMSVoltageSagPeriodPhC*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0a17	*RMSVoltageSwellPeriodPhC*	uint16	0x0000 – 0xffff	RW	0x0000	O

4.9.2.2.11.1 *LineCurrentPhC*

Represents the Phase C, AC line current (Square root of active and reactive current) value at the moment in time the attribute is read, in Amps (A). If the instantaneous current cannot be measured, a value of 0x8000 is returned.

4.9.2.2.11.2 *ActiveCurrentPhC*

Represents the Phase C, AC active/resistive current value at the moment in time the attribute is read, in Amps (A). Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.11.3 *ReactiveCurrentPhC*

Represents the Phase C, AC reactive current value at the moment in time the attribute is read, in Amps (A). Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.11.4 *RMSVoltagePhC*

Represents the most recent RMS voltage reading in Volts (V). If the RMS voltage cannot be measured, a value of 0xffff is returned.

4.9.2.2.11.5 *RMSVoltageMinPhC*

Represents the lowest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of 0xffff until a measurement is made.

4.9.2.2.11.6 *RMSVoltageMaxPhC*

Represents the highest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of 0xffff until a measurement is made.

4.9.2.2.11.7 *RMSCurrentPhC*

Represents the most recent RMS current reading in Amps (A). If the power cannot be measured, a value of 0xffff is returned.

4.9.2.2.11.8 *RMSCurrentMinPhC*

Represents the lowest RMS current value measured in Amps (A). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.11.9 *RMSCurrentMaxPhC*

Represents the highest RMS current value measured in Amps (A). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.11.10 *ActivePowerPhC*

Represents the Phase C, current demand of active power delivered or received at the premises, in Watts (W). Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.11.11 *ActivePowerMinPhC*

Represents the lowest AC power value measured in Watts (W). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.11.12 *ActivePowerMaxPhC*

Represents the highest AC power value measured in Watts (W). After resetting, this attribute will return a value of 0x8000 until a measurement is made.

4.9.2.2.11.13 *ReactivePowerPhC*

Represents the Phase C, current demand of reactive power delivered or received at the premises, in VAr. Positive values indicate power delivered to the premises where negative values indicate power received from the premises.

4.9.2.2.11.14 *ApparentPowerPhC*

Represents the Phase C, current demand of apparent (Square root of active and reactive power) power, in VA.

4.9.2.2.11.15 *PowerFactorPhC*

Contains the Phase C, Power Factor ratio in 1/100ths.

4.9.2.2.11.16 *AverageRMSVoltageMeasurementPeriodPhC*

The Period in seconds that the RMS voltage is averaged over

4.9.2.2.11.17 *AverageRMSOverVoltageCounterPhC*

The number of times the average RMS voltage, has been above the *AverageRMS OverVoltage* threshold since last reset. This counter may be reset by writing zero to the attribute.

4.9.2.2.11.18 *AverageRMSUnderVoltageCounterPhC*

The number of times the average RMS voltage, has been below the *AverageRMS underVoltage* threshold since last reset. This counter may be reset by writing zero to the attribute.

4.9.2.2.11.19 *RMSExtremeOverVoltagePeriodPhC*

The duration in seconds used to measure an extreme over voltage condition.

4.9.2.2.11.20 *RMSExtremeUnderVoltagePeriodPhC*

The duration in seconds used to measure an extreme under voltage condition.

4.9.2.2.11.21 *RMSVoltageSagPeriodPhC*

The duration in seconds used to measure a voltage sag condition.

4.9.2.2.11.22 *RMSVoltageSwellPeriodPhC*

The duration in seconds used to measure a voltage swell condition.

4.9.2.3 Server Commands

4.9.2.3.1 Commands Generated

The command IDs generated by the electrical measurement server cluster are listed in Table 4-41.

Table 4-41. Generated Command ID's for the Electrical Measurement Server

Command Identifier	Description	M/O
0x00	Get Profile Info Response Command	O
0x01	Get Measurement Profile Response Command	O

4.9.2.3.1.1 Get Profile Info Response Command

The Get Profile Info Response Command shall be formatted as illustrated in Figure 4-6.

Figure 4-6. Format of the Get Profile Info Response Command

Octets	1	1	1	Variable
Data Type	uint8	enum8	uint8	Array of attribute IDs (two-byte unsigned values)
Field Name	Profile Count	ProfileIntervalPeriod	MaxNumberOfIntervals	ListOfAttributes

4.9.2.3.1.2 Payload Details

Profile Count: Total number of supported profile.

ProfileIntervalPeriod: Represents the interval or time frame used to capture parameter for profiling purposes. ProfileIntervalPeriod is an enumerated field representing the timeframes listed in Figure 4-7.

Figure 4-7. ProfileIntervalPeriod

Enumerated Value	Time Frame
0	Daily
1	60 minutes
2	30 minutes
3	15 minutes
4	10 minutes
5	7.5 minutes
6	5 minutes
7	2.5 minutes

MaxNumberOfIntervals: Represents the maximum number of intervals the device is capable of returning in one Get Measurement Profile Response command. It is required MaxNumberofIntervals fit within the default Fragmentation ASDU size of 128 bytes, or an optionally agreed upon larger Fragmentation ASDU size supported by both devices as per the application profile supported by the devices.

ListOfAttributes: Represents the list of attributes being profiled.

4.9.2.3.1.3 When Generated

This command is generated when the Client command GetProfileInfo is received.

4.9.2.3.1.4 Get Measurement Profile Response Command

The Get Measurement Profile Response Command shall be formatted as illustrated in Figure 4-8.

Figure 4-8. Format of the Get Measurement Profile Response Command

Octets	4	1	1	1	1	Variable
Data Type	UTC	enum8	enum8	uint8	attribId	Array of Attribute values
Field Name	StartTime	Status	ProfileInter valPeriod	NumberOf IntervalsDelivered	Attribute Id	Intervals

4.9.2.3.1.5 Payload Details

StartTime: 32-bit value (in UTC) representing the end time of the most chronologically recent interval being requested. Example: Data collected from 2:00 PM to 3:00 PM would be specified as a 3:00 PM interval (end time).

Status: Table status enumeration in Table 4-42 lists the valid values returned in the Status field.

Table 4-42. List of Status Valid Values

Status Value	**Description**
0x00	Success
0x01	Attribute Profile not supported
0x02	Invalid Start Time
0x03	More intervals requested than can be returned
0x04	No intervals available for the requested time

ProfileIntervalPeriod: Represents the interval or time frame used to capture parameter for profiling purposes. Refer to table “ProfileIntervalPeriod”.

NumberofIntervalsDelivered: Represents the number of intervals the device is returning. Please note the number of intervals returned in the Get Measurement Profile Response command can be calculated when the packets are received and can replace the usage of this field. The intent is to provide this information as a convenience.

AttributeID: The attribute that has been profiled by the application.

Intervals: Series of interval data captured using the period specified by the ProfileIntervalPeriod field. The content of the interval data depend of the type of information requested using the **AttributeID** field in the Get Measurement Profile Command. Data is organized in a reverse chronological order, the oldest intervals are transmitted first and the newest interval is transmitted last. Invalid intervals should be marked as 0xffff. For scaling and data type use the respective attribute set as defined above in attribute sets.

4.9.2.3.1.6 When Generated

This command is generated when the Client command GetMeasurementProfile is received.

4.9.2.4 Client Commands

4.9.2.4.1 Commands Generated

The command ID's generated by the electrical measurement client cluster are listed in Table 4-43.

Table 4-43. Generated Command IDs for the Electrical Measurement Client

Command Identifier	Description	M/O
0x00	Get Profile Info Command	O
0x01	Get Measurement Profile Command	O

4.9.2.4.1.1 Get Profile Info Command

This command has no payload.

4.9.2.4.1.2 Effect on Receipt

On receipt of this command, the device shall send a Get Profile Info Response Command. A ZCL default response with status UNSUP_CLUSTER_COMMAND shall be returned if command is not supported on the device.

4.9.2.4.1.3 Get Measurement Profile Command

The Get Measurement Profile Command shall be formatted as illustrated in Figure 4-9.

Figure 4-9. Format of the Get Measurement Profile Command

Octets	2	1	1
Data Type	attribId	UTC Time	uint8
Field Name	Attribute ID	Start Time	NumberOfIntervals

4.9.2.4.1.4 Payload Details

Attribute ID: The electricity measurement attribute being profiled.

StartTime: 32-bit value (in UTCTime) used to select an Intervals block from all the Intervals blocks available. The Intervals block returned is the most recent block with its StartTime equal or greater to the one provided.

NumberOfIntervals: Represents the number of intervals being requested. This value can't exceed the size stipulated in the MaxNumberOfIntervals field of Get Profile Info Response Command. If more intervals are requested than can be delivered, the GetMeasurementProfileResponse will return the number of intervals equal to MaxNumberOfIntervals. If fewer intervals available for the time period then only those available are returned.

4.9.2.4.1.5 Effect on Receipt

On receipt of this command, the device shall send a Get Measurement Profile Response Command. A ZCL default response with status UNSUP_CLUSTER_COMMAND shall be returned if command is not supported on the device.

4.10 Electrical Conductivity Measurement

4.10.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The server cluster provides an interface to Electrical Conductivity measurement functionality.

4.10.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

4.10.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	EC	Type 2 (server to client)

4.10.1.3 Cluster Identifiers

Identifier	Name
0x040A	Electrical Conductivity

4.10.2 Server

4.10.2.1 Attributes

Table 4-44. Attributes of the Electrical Conductivity Measurement server cluster

Id	Name	Type	Range	Acc	Def	MO
0x0000	*MeasuredValue*	uint16	*MinMeasuredValue* to *MaxMeasuredValue*	RP	0xffff	M
0x0001	*MinMeasuredValue*	uint16	0x0000 to *MaxMeasuredValue*-1	R	0xffff	M
0x0002	*MaxMeasuredValue*	uint16	*MinMeasuredValue*+1 to 0xfffe	R	0xffff	M

Id	Name	Type	Range	Acc	Def	MO
0x0003	*Tolerance*	uint16	0x0000 to 0x0064	R		O

4.10.2.1.1 *MeasuredValue* Attribute

MeasuredValue represents the Electrical Conductivity in EC or mS/m (milli-Siemens per meter) as follows:

MeasuredValue = 10 x Electrical Conductivity in mS/m. The maximum resolution this format allows is 0.1.

A *MeasuredValue* of 0xffff SHALL indicate an unknown value, otherwise the range SHALL be as described in 4.1.3.

MeasuredValue is updated continuously as new measurements are made.

4.10.2.1.2 *MinMeasuredValue* Attribute

The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that is capable of being measured. A *MinMeasuredValue* of 0xffff indicates that the minimum value is not defined. See 4.1.3 for more details.

4.10.2.1.3 *MaxMeasuredValue* Attribute

The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that is capable of being measured. A *MaxMeasuredValue* of 0xffff indicates that the maximum value is not defined. See 4.1.3 for more details.

4.10.2.1.4 *Tolerance* Attribute

See 4.1.3.

4.10.2.2 Commands

No cluster specific commands are generated or received by the server cluster.

4.10.3 Client

The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

4.11 pH Measurement

4.11.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The server cluster provides an interface to pH measurement functionality.

4.11.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

4.11.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	PH	Type 2 (server to client)

4.11.1.3 Cluster Identifiers

Identifier	Name
0x0409	pH Measurement

4.11.2 Server

4.11.2.1 Attributes

Table 4-45. Attributes of the pH Measurement server cluster

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*MeasuredValue*	uint16	*MinMeasuredValue* to *MaxMeasuredValue*	RP	0xffff	M
0x0001	*MinMeasuredValue*	uint16	0x0000 to *MaxMeasuredValue*-1	R	0xffff	M
0x0002	*MaxMeasuredValue*	uint16	*MinMeasuredValue*+1 to 0x0578	R	0xffff	M
0x0003	*Tolerance*	uint16	0x0000 to 0x00c8	R		O

4.11.2.1.1 *MeasuredValue* Attribute

MeasuredValue represents the pH with no units as follows: *MeasuredValue* = 100 x pH.

Where 0.00 <= pH <= 14.00, corresponding to a *MeasuredValue* in the range 0x0000 to 0x0578. The maximum resolution this format allows is 0.01, this is to accommodate certain applications where such resolution is necessary.

A *MeasuredValue* of 0xffff SHALL indicate an unknown value, otherwise the range SHALL be as described in 4.1.3.

MeasuredValue is updated continuously as new measurements are made.

4.11.2.1.2 *MinMeasuredValue* Attribute

The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that is capable of being measured. A *MinMeasuredValue* of 0xffff indicates that the minimum value is not defined. See 4.1.3 for more details.

4.11.2.1.3 *MaxMeasuredValue* Attribute

The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that is capable of being measured. A *MaxMeasuredValue* of 0xffff indicates that the maximum value is not defined. See 4.1.3 for more details.

4.11.2.1.4 *Tolerance* Attribute

See 4.1.3.

4.11.2.2 Commands

No cluster specific commands are generated or received by the server cluster.

4.11.3 Client

The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

4.12 Wind Speed Measurement

4.12.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The server cluster provides an interface to Wind Speed measurement functionality.

4.12.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

4.12.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	WSPD	Type 2 (server to client)

4.12.1.3 Cluster Identifiers

Identifier	Name
0x040b	Wind Speed Measurement

4.12.2 Server

4.12.2.1 Attributes

Table 4-46. Attributes of the Wind Speed Measurement server cluster

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*MeasuredValue*	uint16	*MinMeasuredValue* to *MaxMeasuredValue*	RP	0xffff	M
0x0001	*MinMeasuredValue*	uint16	0x0000 to *MaxMeasuredValue*-1	R	0xffff	M
0x0002	*MaxMeasuredValue*	uint16	*MinMeasuredValue*+1 to 0xfffe	R	0xffff	M

Id	Name	Type	Range	Acc	Def	M/O
0x0003	*Tolerance*	uint16	0x0000 to 0x0308	R		O

4.12.2.1.1 *MeasuredValue* Attribute

MeasuredValue represents the Wind Speed in m/s (meters per second) as follows:

MeasuredValue = 100 x Wind Speed in m/s. The maximum resolution this format allows is 0.01.

A *MeasuredValue* of 0xffff SHALL indicate an unknown value, otherwise the range SHALL be as described in 4.1.3.

MeasuredValue is updated continuously as new measurements are made.

4.12.2.1.2 *MinMeasuredValue* Attribute

The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that is capable of being measured. A *MinMeasuredValue* of 0xffff indicates that the minimum value is not defined. See 4.1.3 for more details.

4.12.2.1.3 *MaxMeasuredValue* Attribute

The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that is capable of being measured. A *MaxMeasuredValue* of 0xffff indicates that the maximum value is not defined. See 4.1.3 for more details.

4.12.2.1.4 *Tolerance* Attribute

See 4.1.3.

4.12.2.2 Commands

No cluster specific commands are generated or received by the server cluster.

4.12.3 Client

The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

4.13 Concentration Measurement

4.13.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The server cluster provides an interface to concentration measurement functionality. The measurement is reportable and may be configured for reporting. Concentration measurements include, but are not limited to, levels in gases, such as CO, CO2, and ethylene, or in fluids and solids, such as dissolved oxygen, chemicals & pesticides.

4.13.1.1 Revision History

Rev	Description

1	mandatory global *ClusterRevision* attribute added

4.13.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	CONC	Type 2 (server to client)

4.13.1.3 Cluster Identifiers

The table below is a list of Cluster Ids that conform to this specification. More than one ambient substance may be supported by the same Cluster Id (e.g. Water and Alcohol). This would make the Cluster Id generic to the ambient substance. A new Cluster Id may also be added that is limited to a single ambient substance to provide more specific self-description. If both generic and specific Cluster Ids appear on an endpoint, then a single instance of the cluster exists on the endpoint, and either Cluster Id can be used to access the cluster interface.

Cluster Id	Substance Measured	Ambient Substance	Units	Notes (not normative or prescribed)
0x040c	Carbon Monoxide (CO)	Air	Volume	
0x040d	Carbon Dioxide (CO2)	Air	Volume	
0x040e	Ethylene (CH2)	Air	Volume	
0x040f	Ethylene Oxide (C2H4O)	Air	Volume	
0x0410	Hydrogen (H)	Air	Volume	
0x0411	Hydrogen Sulfide (H2S)	Air	Volume	
0x0412	Nitric Oxide (NO)	Air	Volume	
0x0413	Nitrogen Dioxide (NO2)	Air	Volume	
0x0414	Oxygen (O2)	Air	Volume	
0x0415	Ozone (O3)	Air	Volume	
0x0416	Sulfur Dioxide (SO2)	Air	Volume	
0x0417	Dissolved Oxygen (DO)	Water	Mass	
0x0418	Bromate	Drinking Water	Volume	typical range example: not detected to 3.6 PPB typical value example:1.79 PPB
0x0419	Chloramines	Drinking Water	Volume	typical range example: 0.9 to 3.8 PPM typical value example: 2.87 PPM
0x041a	Chlorine	Drinking Water	Volume	typical range example: 0.1 to 2.4 PPM typical value example: 1.28 PPM

0x041b	Fecal coliform & E. Coli	Drinking Water	Volume	Percent of positive samples typical value example: 0
0x041b	Fluoride	Drinking Water	Volume	typical range example: 0 to 100 PPM typical value example: 0.72 PPM
0x041d	Haloacetic Acids	Drinking Water	Volume	typical range example: Not Detected to 20 PPB typical value example: 14 PPB
0x041e	Total Trihalomethanes	Drinking Water	Volume	typical range example: 0 to 100 PPB typical value example: 44 PPB
0x041f	Total Coliform Bacteria	Drinking Water	Volume	Percent of positive samples typical range example: 0 to 100% typical value example: 1.33%
0x0420	Turbidity	Drinking Water	Volume	Cloudiness of particles in water where an average person would notice a 5 or higher typical range example: 0 to 10 typical value example: 0.18
0x0421	Copper	Drinking Water	Volume	typical range example: 0 to 10 PPM typical value example: 0.191 PPM
0x0422	Lead	Drinking Water	Volume	typical range example: 0 to 10 PPB typical value example: 3.2 PPB
0x0423	Manganese	Drinking Water	Volume	typical range example: 0 to 1000 PPB typical value example: 31PPB
0x0424	Sulfate	Drinking Water	Volume	typical range example: 0 to 1000 PPM typical value example: 36 PPM
0x0425	Bromodichloromethane	Drinking Water	Volume	typical range example: 0 to 1000 PPB typical value example: 9.6 PPB
0x0426	Bromoform	Drinking Water	Volume	typical range example: 0 to 1000 PPB typical value example: 1.1 PPB
0x0427	Chlorodibromomethane	Drinking Water	Volume	typical range example: 0 to 1000 PPB typical value example: 6.4 PPB
0x0428	Chloroform	Drinking Water	Volume	typical range example: 0 to 1000 PPB typical value example: 8.0 PPB
0x0429	Sodium	Drinking Water	Volume	typical range example: 0 to 1000 PPM typical value example: 27 PPM
0x042A	PM2.5	Air	Volume	Particulate Matter 2.5 microns or less
0x042B	Formaldehyde	Air	Volume	

4.13.2 Server

4.13.2.1 Attributes

Table 4-47. Attributes of the Concentration Measurement server cluster

Id	Name	Type	Range	Acc	Def	MO
0x0000	*MeasuredValue*	single	*MinMeasuredValue* to *MaxMeasuredValue*	RP	NaN*	M
0x0001	*MinMeasuredValue*	single	0 <= *value* < *MaxMeasuredValue*	R	NaN*	M
0x0002	*MaxMeasuredValue*	single	*MinMeasuredValue* < *value* <= 1	R	NaN*	M
0x0003	*Tolerance*	single	*MS*	R	*MS*	O

* see Not a Number: Chapter 2 for data type default and invalid values

4.13.2.1.1 *MeasuredValue* Attribute

MeasuredValue represents the concentration as a fraction of 1 (one).

A value of NaN indicates that the concentration measurement is unknown or outside the valid range.

MinMeasuredValue and *MaxMeasuredValue* define the valid range for *MeasuredValue*.

MeasuredValue is updated continuously as new measurements are made.

4.13.2.1.2 *MinMeasuredValue* Attribute

The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that is capable of being measured. A *MinMeasuredValue* of NaN indicates that the *MinMeasuredValue* is not defined. See 4.1.3 for more details.

4.13.2.1.3 *MaxMeasuredValue* Attribute

The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that is capable of being measured. A *MaxMeasuredValue* of NaN indicates that the *MaxMeasuredValue* is not defined. See 4.1.3 for more details.

4.13.2.1.4 *Tolerance* Attribute

See 4.1.3.

4.13.2.2 Commands

No cluster specific commands are generated or received by the server cluster.

4.13.3 Client

The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

CHAPTER 5 LIGHTING

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

5.1 General Description

5.1.1 Introduction

The clusters specified in this document are for use typically in lighting applications, but MAY be used in any application domain.

5.1.2 Terms

Ballast Factor: A measure of the light output (lumens) of a ballast and lamp combination in comparison to an ANSI standard ballast operated with the same lamp. Multiply the ballast factor by the rated lumens of the lamp to get the light output of the lamp/ballast combination.

HSV: Hue, Saturation, Value. A color space, also known as HSB (Hue, Saturation, Brightness). This is a well-known transformation of the RGB (Red, Green, Blue) color space. For more information see e.g., http://en.wikipedia.org/wiki/HSV_color_space.

Illuminance: The density of incident luminous flux on a surface. Illuminance is the standard metric for lighting levels, and is measured in lux (lx).

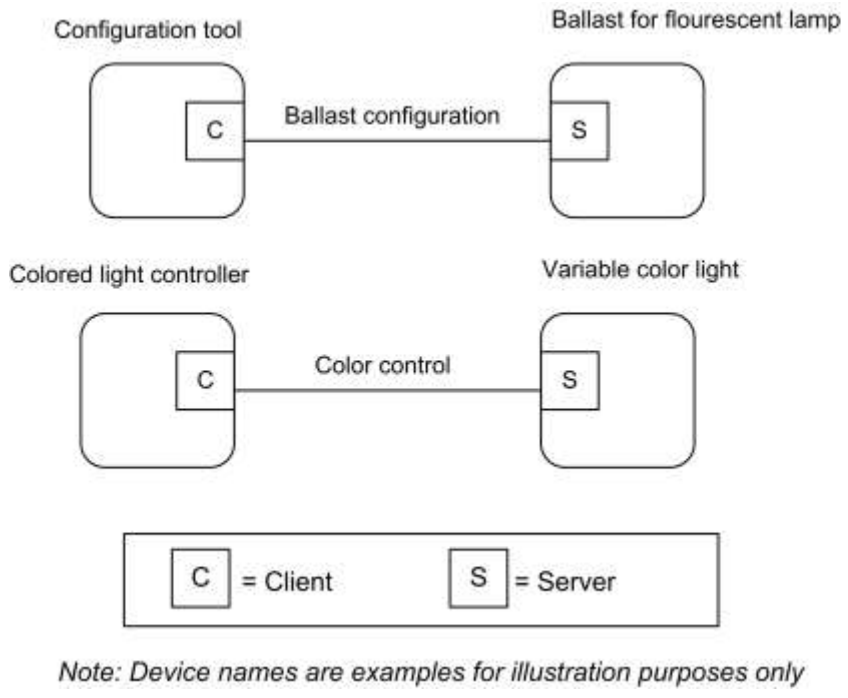
5.1.3 Cluster List

This section lists the clusters specified in this document, and gives examples of typical usage for the purpose of clarification. The clusters specified in this document are listed in Table 5-1.

Table 5-1. Clusters Specified for the Lighting Functional Domain

ID	Cluster Name	Description
0x0300	Color Control	Attributes and commands for controlling the color of a color-capable light.
0x0301	Ballast Configuration	Attributes and commands for configuring a lighting ballast

Figure 5-1. Typical Usage of Ballast Configuration and Color Control Clusters



5.2 Color Control Cluster

5.2.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface for changing the color of a light. Color is specified according to the Commission Internationale de l'Éclairage (CIE) specification CIE 1931 Color Space, [I1]. Color control is carried out in terms of x,y values, as defined by this specification.

Additionally, color MAY optionally be controlled in terms of color temperature, or as hue and saturation values based on optionally variable RGB and W color points. It is recommended that the hue and saturation are interpreted according to the HSV (aka HSB) color model.

Control over luminance is not included, as this is provided by means of the Level Control for Lighting cluster of the General library (see Chapter 3). It is recommended that the level provided by this cluster be interpreted as representing a proportion of the maximum intensity achievable at the current color.

5.2.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added; CCB 2028
2	added *Options* attribute, CCB 2085 2104 2124 2230; ZLO 1.0

5.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	CC	Type 1 (client to server)

5.2.1.3 Cluster Identifiers

Identifier	Name
0x0300	Color Control

5.2.2 Server

5.2.2.1 Dependencies

5.2.2.1.1 Coupling color temperature to Level Control[67]

If the *Level Control for Lighting* cluster identifier 0x0008 is supported on the same endpoint as the *Color Control* cluster and color temperature is supported, it is possible to couple changes in the current level to the color temperature.

The *CoupleColorTempToLevel* bit of the *Options* attribute of the *Level Control* cluster indicates whether the color temperature is to be linked with the *CurrentLevel* attribute in the *Level Control* cluster.

If the *CoupleColorTempToLevel* bit of the *Options* attribute of the *Level Control* cluster is equal to 1 and the *ColorMode* or *EnhancedColorMode* attribute is set to 0x02 (*color temperature*) then a change in the *CurrentLevel* attribute SHALL affect the *ColorTemperatureMireds* attribute. This relationship is manufacturer specific, with the qualification that the maximum value of the *CurrentLevel* attribute SHALL correspond to a *ColorTemperatureMired* attribute value equal to the *CoupleColorTempToLevelMinMireds* attribute. This relationship is one-way so a change to the *ColorTemperatureMireds* attribute SHALL NOT have any effect on the *CurrentLevel* attribute.

In order to simulate the behavior of an incandescent bulb, a low value of the *CurrentLevel* attribute SHALL be associated with a high value of the *ColorTemperatureMireds* attribute (i.e., a low value of color temperature in kelvins).

If the *CoupleColorTempToLevel* bit of the *Options* attribute of the *Level Control* cluster is equal to 0, there SHALL be no link between color temperature and current level.

5.2.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 5-2.

Table 5-2. Hue Control Attribute Sets

Attribute Set Identifier	Description
0x000, 0x400	Color Information

[67] ZLO 1.0

0x001	Defined Primaries Information
0x002	Additional Defined Primaries Information
0x003	Defined Color Point Settings

5.2.2.2.1 Color Information Attribute Set

The Color Information attribute set contains the attributes summarized below.

Table 5-3. Attributes of the Color Information Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0000	CurrentHue	uint8	0x00 – 0xfe	RP	0x00	M^0
0x0001	CurrentSaturation	uint8	0x00 – 0xfe	RPS	0x00	M^0
0x0002	RemainingTime	uint16	0x0000 – 0xfffe	R	0x00	O
0x0003	CurrentX	uint16	0x0000 - 0xfeff	RP Scene	0x616b (0.381)	M^3
0x0004	CurrentY	uint16	0x0000 - 0xfeff	RPS	0x607d (0.377)	M^3
0x0005	DriftCompensation	enum8	0x00 – 0x04	R	-	O
0x0006	CompensationText	string	0 to 254 chars	R	-	O
0x0007	ColorTemperatureMireds	uint16	0x0000 - 0xfeff	RPS[68]	0x00fa (4000K)	M^4
0x0008	ColorMode	enum8	0x00 – 0x02	R	0x01	M
0x000f	Options[69]	map8		RW	0x00	M
0x4000	*EnhancedCurrentHue*	uint16	0x0000 – 0xffff	RS	0x0000	M^1
0x4001	*EnhancedColorMode*	enum8	0x00 – 0xff	R	0x01[70]	M
0x4002	*ColorLoopActive*	uint8	0x00 – 0xff	RS	0x00	M^2
0x4003	*ColorLoopDirection*	uint8	0x00 – 0xff	RS	0x00	M^2
0x4004	*ColorLoopTime*	uint16	0x0000 – 0xffff	RS	0x0019	M^2
0x4005	*ColorLoopStartEnhancedHue*	uint16	0x0000 – 0xffff	R	0x2300	M^2
0x4006	*ColorLoopStoredEnhancedHue*	uint16	0x0000 – 0xffff	R	0x0000	M^2
0x400a	*ColorCapabilities*	map16	0x0000 – 0x001f	R	0x0000	M
0x400b	*ColorTempPhysicalMinMireds*	uint16	0x0000 – 0xfeff	R	0x0000	M^4
0x400c	*ColorTempPhysicalMaxMireds*	uint16	0x0000 – 0xfeff	R	0xfeff	M^4
0x400d	*CoupleColorTempToLevelMinMireds*[71]	uint16	*ColorTempPhysicalMinMireds* to *ColorTempPhysicalMaxMireds*	R	MS	M^{4*}
0x4010	*StartUpColorTemperatureMireds*[72]	uint16	0x0000-0xffff	RW	MS	M^{4*}

68 ZLO 1.0
69 CCB 2085
70 CCB 2124
71 ZLO 1.0
72 ZLO 1.0

M^i = Mandatory if bit *i* of the *ColorCapabilities* attribute is equal to 1, otherwise optional.

* Mandatory if *ColorTemperatureMireds is supported.*

5.2.2.2.1.1 *CurrentHue* Attribute

The *CurrentHue* attribute contains the current hue value of the light. It is updated as fast as practical during commands that change the hue.

The hue in degrees SHALL be related to the CurrentHue attribute by the relationship Hue = CurrentHue x 360 / 254 (CurrentHue in the range 0 - 254 inclusive)

If this attribute is implemented then the *CurrentSaturation* and *ColorMode* attributes SHALL also be implemented.

5.2.2.2.1.2 *CurrentSaturation* Attribute

The *CurrentSaturation* attribute holds the current saturation value of the light. It is updated as fast as practical during commands that change the saturation.

The saturation SHALL be related to the CurrentSaturation attribute by the relationship Saturation = CurrentSaturation/254 (CurrentSaturation in the range 0 - 254 inclusive)

If this attribute is implemented then the *CurrentHue* and *ColorMode* attributes SHALL also be implemented.

5.2.2.2.1.3 *RemainingTime* Attribute

The *RemainingTime* attribute holds the time remaining, in 1/10ths of a second, until the currently active command will be complete.

5.2.2.2.1.4 *CurrentX* Attribute

The *CurrentX* attribute contains the current value of the normalized chromaticity value x, as defined in the CIE xyY Color Space. It is updated as fast as practical during commands that change the color.

The value of x SHALL be related to the *CurrentX* attribute by the relationship

x = *CurrentX* / 65536 (*CurrentX* in the range 0 to 65279 inclusive)

5.2.2.2.1.5 *CurrentY* Attribute

The *CurrentY* attribute contains the current value of the normalized chromaticity value y, as defined in the CIE xyY Color Space. It is updated as fast as practical during commands that change the color.

The value of y SHALL be related to the *CurrentY* attribute by the relationship

y = *CurrentY* / 65536 (*CurrentY* in the range 0 to 65279 inclusive)

5.2.2.2.1.6 *DriftCompensation* Attribute

The *DriftCompensation* attribute indicates what mechanism, if any, is in use for compensation for color/intensity drift over time. It SHALL be one of the non-reserved values in Table 5-4.

Table 5-4. Values of the *DriftCompensation* Attribute

Attribute Value	Description
0x00	None
0x01	Other / Unknown
0x02	Temperature monitoring

0x03	Optical luminance monitoring and feedback
0x04	Optical color monitoring and feedback

5.2.2.2.1.7 *CompensationText* Attribute

The *CompensationText* attribute holds a textual indication of what mechanism, if any, is in use to compensate for color/intensity drift over time.

5.2.2.2.1.8 *ColorTemperatureMireds* Attribute

The *ColorTemperatureMireds* attribute contains a scaled inverse of the current value of the color temperature. The unit of *ColorTemperatureMireds* is the mired (micro reciprocal degree), AKA mirek (micro reciprocal kelvin). It is updated as fast as practical during commands that change the color.

The color temperature value in kelvins SHALL be related to the *ColorTemperatureMireds* attribute in mireds by the relationship

Color temperature in kelvins = 1,000,000 / *ColorTemperatureMireds*, where *ColorTemperatureMireds* is in the range 1 to 65279 mireds inclusive, giving a color temperature range from 1,000,000 kelvins to 15.32 kelvins.

The value *ColorTemperatureMireds* = 0x0000 indicates an undefined value. The value *ColorTemperatureMireds* = 0xffff indicates an invalid value.

If this attribute is implemented then the *ColorMode* attribute SHALL also be implemented.

5.2.2.2.1.9 *ColorMode* Attribute

The *ColorMode* attribute indicates which attributes are currently determining the color of the device. If either the *CurrentHue* or *CurrentSaturation* attribute is implemented, this attribute SHALL also be implemented, otherwise it is optional.

The value of the *ColorMode* attribute cannot be written directly - it is set upon reception of any command in section 5.2.2.3 to the appropriate mode for that command.

Table 5-5. Values of the *ColorMode* Attribute

Attribute Value	Attributes that Determine the Color
0x00	*CurrentHue* and *CurrentSaturation*
0x01	*CurrentX* and *CurrentY*
0x02	*ColorTemperatureMireds*

5.2.2.2.1.10 *Options* Attribute[73]

The *Options* attribute is meant to be changed only during commissioning. The *Options* attribute is a bitmap that determines the default behavior of some cluster commands. Each command that is dependent on the *Options* attribute SHALL first construct a temporary Options bitmap that is in effect during the command processing. The temporary Options bitmap has the same format and meaning as the *Options* attribute, but includes any bits that may be overridden by command fields.

Below is the format and description of the *Options* attribute and temporary Options bitmap and the effect on dependent commands.

[73] CCB 2085

Table 5-6. *Options* Attribute

Bit	Name	Values & Summary
0	ExecuteIfOff	0 – Do not execute command if the On/Off cluster, OnOff attribute is 0x00 (FALSE) 1 – Execute command if the On/Off cluster, OnOff attribute is 0x00 (FALSE)

ExecuteIfOff: Command execution SHALL NOT continue beyond the *Options* processing if all of these criteria are true:

- The On/Off cluster exists on the same endpoint as this cluster.
- The *OnOff* attribute of the On/Off cluster, on this endpoint, is 0x00 (FALSE).
- The value of the ExecuteIfOff bit is 0.

5.2.2.2.1.11 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 the ZLL test specification 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.

5.2.2.2.1.12 EnhancedColorMode Attribute

The EnhancedColorMode attribute specifies which attributes are currently determining the color of the device, as detailed in Table 5-7.

Table 5-7. Values of the *EnhancedColorMode* Attribute

Attribute Value	Attributes That Determine the Color
0x00	CurrentHue and CurrentSaturation
0x01	*CurrentX* and *CurrentY*
0x02	*ColorTemperatureMireds*
0x03	*EnhancedCurrentHue* and *CurrentSaturation*

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.

5.2.2.2.1.13 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.

5.2.2.2.1.14 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.

5.2.2.2.1.15 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).

5.2.2.2.1.16 ColorLoopStartEnhancedHue Attribute

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

5.2.2.2.1.17 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.

5.2.2.2.1.18 ColorCapabilities Attribute

The ColorCapabilities attribute specifies the color capabilities of the device supporting the color control cluster, as illustrated in Table 5-8. 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 5-8. Bit Values of the *ColorCapabilities* Attribute

Value	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* *Stop move step*
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* *Stop move step*
2	Color loop supported **Note:** enhanced hue must also be supported.	*ColorLoopActive* *ColorLoopDirection* *ColorLoopTime* *ColorLoopStartEnhancedHue* *ColorLoopStoredEnhancedHue*	*Color loop set*

Value	Description	Related Attributes	Mandatory Commands
3	XY attributes supported	*CurrentX* *CurrentY*	*Move to color* *Move color* *Step color* *Stop move step*
4	Color temperature supported	*ColorTemperatureMireds* *ColorTempPhysicalMinMireds* *ColorTempPhysicalMaxMireds*	*Move to color temperature* *Move color temperature* *Step color temperature* *Stop move step*

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.

5.2.2.2.1.19 ColorTempPhysicalMinMireds Attribute

The ColorTempPhysicalMinMireds attribute indicates the minimum mired value supported by the hardware. ColorTempPhysicalMinMireds corresponds to the maximum color temperature in kelvins supported by the hardware. ColorTempPhysicalMinMireds ≤ ColorTemperatureMireds.

5.2.2.2.1.20 ColorTempPhysicalMaxMireds Attribute

The ColorTempPhysicalMaxMireds attribute indicates the maximum mired value supported by the hardware. ColorTempPhysicalMaxMireds corresponds to the minimum color temperature in kelvins supported by the hardware. ColorTemperatureMireds ≤ ColorTempPhysicalMaxMireds.

5.2.2.2.1.21 CoupleColorTempToLevelMinMireds Attribute[74]

The *CoupleColorTempToLevelMinMireds* attribute specifies a lower bound on the value of the *ColorTemperatureMireds* attribute for the purposes of coupling the *ColorTemperatureMireds* attribute to the *CurrentLevel* attribute when the *CoupleColorTempToLevel* bit of the *Options* attribute of the *Level Control* cluster is equal to 1. When coupling the *ColorTemperatureMireds* attribute to the *CurrentLevel* attribute, this value SHALL correspond to a *CurrentLevel* value of 0xfe (100%).

This attribute SHALL be set such that the following relationship exists:

$$ColorTempPhysicalMinMireds \le CoupleColorTempToLevelMinMireds \le ColorTemperatureMireds$$

Note that since this attribute is stored as a micro reciprocal degree (mired) value (i.e. color temperature in kelvins = 1,000,000 / *CoupleColorTempToLevelMinMireds*), the *CoupleColorTempToLevel-MinMireds* attribute corresponds to an upper bound on the value of the color temperature in kelvins supported by the device.

5.2.2.2.1.22 StartUpColorTemperatureMireds Attribute[75]

The *StartUpColorTemperatureMireds* attribute SHALL define the desired startup color temperature value a lamp SHALL use when it is supplied with power and this value SHALL be reflected in the *ColorTemperatureMireds* attribute. In addition, the *ColorMode* and *EnhancedColorMode* attributes SHALL be set to 0x02 (*color temperature*). The values of the *StartUpColorTemperatureMireds* attribute are listed in the table below.

[74] ZLO 1.0
[75] ZLO 1.0

Table 5-9. Values of the *StartUpColorTemperatureMireds* attribute

Value	Action on power up
0x0000 – 0xffef	Set the *ColorTemperatureMireds* attribute to this value.
0xffff	Set the *ColorTemperatureMireds* attribute to its previous value.

5.2.2.2.2 Defined Primaries Information Attribute Set

The Defined Primaries Information attribute set contains the attributes summarized in Table 5-10.

Table 5-10. Defined Primaries Information Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0010	NumberOfPrimaries	uint8	0x00 – 0x06	R	-	M
0x0011	Primary1X	uint16	0x0000 – 0xfeff	R	-	M^0
0x0012	Primary1Y	uint16	0x0000 – 0xfeff	R	-	M^0
0x0013	Primary1Intensity	uint8	0x00 - 0xff	R	-	M^0
0x0014	Reserved	-	-	-	-	-
0x0015	Primary2X	uint16	0x0000 – 0xfeff	R	-	M^1
0x0016	Primary2Y	uint16	0x0000 – 0xfeff	R	-	M^1
0x0017	Primary2Intensity	uint8	0x0000- 0xff	R	-	M^1
0x0018	Reserved	-	-	-	-	-
0x0019	Primary3X	uint16	0x0000 – 0xfeff	R	-	M^2
0x001a	Primary3Y	uint16	0x0000 – 0xfeff	R	-	M^2
0x001b	Primary3Intensity	uint8	0x00 - 0xff	R	-	M^2

M^i = Mandatory if the value of the *NumberOfPrimaries* attribute is greater than *i*, otherwise optional.

5.2.2.2.2.1 *NumberOfPrimaries* Attribute

The *NumberOfPrimaries* attribute contains the number of color primaries implemented on this device. A value of 0xff SHALL indicate that the number of primaries is unknown.

Where this attribute is implemented, the attributes below for indicating the “x” and “y” color values of the primaries SHALL also be implemented for each of the primaries from 1 to *NumberOfPrimaries*, without leaving gaps. Implementation of the *Primary1Intensity* attribute and subsequent intensity attributes is optional.

5.2.2.2.2.2 *Primary1X* Attribute

The *Primary1X* attribute contains the normalized chromaticity value x for this primary, as defined in the CIE xyY Color Space.

The value of x SHALL be related to the *Primary1X* attribute by the relationship x = *Primary1X* / 65536 (*Primary1X* in the range 0 to 65279 inclusive)

5.2.2.2.2.3 *Primary1Y* Attribute

The *Primary1Y* attribute contains the normalized chromaticity value y for this primary, as defined in the CIE xyY Color Space.

The value of y SHALL be related to the *Primary1Y* attribute by the relationship

y = *Primary1Y* / 65536 (*Primary1Y* in the range 0 to 65279 inclusive)

5.2.2.2.2.4 *Primary1Intensity* Attribute

The *Primary1intensity* attribute contains a representation of the maximum intensity of this primary as defined in the Dimming Light Curve in the Ballast Configuration cluster (see 5.3), normalized such that the primary with the highest maximum intensity contains the value 0xfe.

A value of 0xff SHALL indicate that this primary is not available.

5.2.2.2.2.5 Remaining Attributes

The *Primary2X, Primary2Y, Primary2Intensity, Primary3X, Primary3Y and Primary3Intensity* attributes are used to represent the capabilities of the 2nd and 3rd primaries, where present, in the same way as for the *Primary1X, Primary1Y and Primary1Intensity* attributes.

5.2.2.2.3 Additional Defined Primaries Information Attribute Set

The Additional Defined Primaries Information attribute set contains the attributes summarized in Table 5-11.

Table 5-11. Additional Defined Primaries Information Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0020	Primary4X	uint16	0x0000 – 0xfeff	R	-	M^3
0x0021	Primary4Y	uint16	0x0000 – 0xfeff	R	-	M^3
0x0022	Primary4Intensity	uint8	0x00 – 0xff	R	-	M^3
0x0023	Reserved	-	-	-	-	-
0x0024	Primary5X	uint16	0x0000 – 0xfeff	R	-	M^4
0x0025	Primary5Y	uint16	0x0000 – 0xfeff	R	-	M^4
0x0026	Primary5Intensity	uint8	0x00 – 0xff	R	-	M^4
0x0027	Reserved	-	-	-	-	-
0x0028	Primary6X	uint16	0x0000 – 0xfeff	R	-	M^5
0x0029	Primary6Y	uint16	0x0000 – 0xfeff	R	-	M^5
0x002a	Primary6Intensity	uint8	0x00 – 0xff	R	-	M^5

M^i = Mandatory if the value of the *NumberOfPrimaries* attribute is greater than *i*, otherwise optional.

5.2.2.2.3.1 Attributes

The *Primary4X*, *Primary4Y*, *Primary4Intensity*, *Primary5X*, *Primary5Y*, *Primary5Intensity*, *Primary6X*, *Primary6Y* and *Primary6Intensity* attributes represent the capabilities of the 4th, 5th and 6th primaries, where present, in the same way as the *Primary1X*, *Primary1Y* and *Primary1Intensity* attributes.

5.2.2.2.4 Defined Color Points Settings Attribute Set

The Defined Color Points Settings attribute set contains the attributes summarized in Table 5-12.

Table 5-12. Defined Color Points Settings Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0030	WhitePointX	uint16	0x0000 – 0xfeff	RW	-	O
0x0031	WhitePointY	uint16	0x0000 – 0xfeff	RW	-	O
0x0032	ColorPointRX	uint16	0x0000 – 0xfeff	RW	-	O
0x0033	ColorPointRY	uint16	0x0000 – 0xfeff	RW	-	O
0x0034	ColorPointRIntensity	uint8	0x00 – 0xff	RW	-	O
0x0035	Reserved	-	-	-	-	-
0x0036	ColorPointGX	uint16	0x0000 – 0xfeff	RW	-	O
0x0037	ColorPointGY	uint16	0x0000 – 0xfeff	RW	-	O
0x0038	ColorPointGIntensity	uint8	0x00 – 0xff	RW	-	O
0x0039	Reserved	-	-	-	-	-
0x003a	ColorPointBX	uint16	0x0000 – 0xfeff	RW	-	O
0x003b	ColorPointBY	uint16	0x0000 – 0xfeff	RW		O
0x003c	ColorPointBIntensity	uint8	0x00 – 0xff	RW		O

5.2.2.2.4.1 *WhitePointX* Attribute

The *WhitePointX* attribute contains the normalized chromaticity value x, as defined in the CIE xyY Color Space, of the current white point of the device.

The value of x SHALL be related to the *WhitePointX* attribute by the relationship

x = *WhitePointX* / 65536 (*WhitePointX* in the range 0 to 65279 inclusive)

5.2.2.2.4.2 *WhitePointY* Attribute

The *WhitePointY* attribute contains the normalized chromaticity value y, as defined in the CIE xyY Color Space, of the current white point of the device.

The value of y SHALL be related to the *WhitePointY* attribute by the relationship

y = *WhitePointY* / 65536 (*WhitePointY* in the range 0 to 65279 inclusive)

5.2.2.2.4.3 *ColorPointRX* Attribute

The *ColorPointRX* attribute contains the normalized chromaticity value x, as defined in the CIE xyY Color Space, of the red color point of the device.

The value of x SHALL be related to the *ColorPointRX* attribute by the relationship

x = *ColorPointRX* / 65536 (*ColorPointRX* in the range 0 to 65279 inclusive)

5.2.2.2.4.4 *ColorPointRY* Attribute

The *ColorPointRY* attribute contains the normalized chromaticity value y, as defined in the CIE xyY Color Space, of the red color point of the device.

The value of y SHALL be related to the *ColorPointRY* attribute by the relationship

y = *ColorPointRY* / 65536 (*ColorPointRY* in the range 0 to 65279 inclusive)

5.2.2.2.4.5 *ColorPointRIntensity* Attribute

The *ColorPointRIntensity* attribute contains a representation of the relative intensity of the red color point as defined in the Dimming Light Curve in the Ballast Configuration cluster (see 5.3), normalized such that the color point with the highest relative intensity contains the value 0xfe.

A value of 0xff SHALL indicate an invalid value.

5.2.2.2.4.6 Remaining Attributes

The *ColorPointGX*, *ColorPointGY*, *ColorPointGIntensity*, *ColorPointBX*, *ColorPointBY* and, *ColorPointBIntensity* attributes are used to represent the chromaticity values and intensities of the green and blue color points, in the same way as for the *ColorPointRX*, *ColorPointRY* and *ColorPointRIntensity* attributes.

If any one of these red, green or blue color point attributes is implemented then they SHALL all be implemented.

5.2.2.3 Commands Received

The command IDs for the Color Control cluster are listed in Table 5-13.

Table 5-13. Command IDs for the Color Control Cluster

Command Identifier	Description	M/O
0x00	Move to Hue	M^0
0x01	Move Hue	M^0
0x02	Step Hue	M^0
0x03	Move to Saturation	M^0
0x04	Move Saturation	M^0
0x05	Step Saturation	M^0
0x06	Move to Hue and Saturation	M^0
0x07	Move to Color	M^3
0x08	Move Color	M^3

Command Identifier	Description	M/O
0x09	Step Color	M[3]
0x0a	Move to Color Temperature	M[4]
0x40	Enhanced Move to Hue	M[1]
0x41	Enhanced Move Hue	M[1]
0x42	Enhanced Step Hue	M[1]
0x43	Enhanced Move to Hue and Saturation	M[1]
0x44	Color Loop Set	M[2]
0x47	Stop Move Step	M[0,1,3,4]
0x4b	Move Color Temperature	M[4]
0x4c	Step Color Temperature	M[4]

M^i = Mandatory if bit *i* of the *ColorCapabilities* attribute is equal to 1, otherwise optional.

5.2.2.3.1 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.

For the move hue command, the Rate field specifies the rate of movement in steps per second. A step is a change in the device’s hue of one unit. If the move mode field is equal to 0x01 (Up) or 0x03 (Down) and the Rate field has a value of zero, the command has no effect and a default response command (see 2.4.12) is sent in response, with the status code set to INVALID_FIELD.

For the move to color temperature command, if the target color specified is not achievable by the hardware then the color temperature SHALL be clipped at the physical minimum or maximum achievable, depending on the color temperature transition, when the device reaches that color temperature (which MAY be before the requested transition time), and a ZCL default response command SHALL be generated, where not disabled, with a status code equal to SUCCESS.

5.2.2.3.2 Note on Change of ColorMode

The first action taken when any one of these commands is received is to change the *ColorMode* attribute to the appropriate value for the command (see individual commands). Note that, when moving from one color mode to another (e.g., *CurrentX*/*CurrentY* to *CurrentHue*/*CurrentSaturation*), the starting color for the command is formed by calculating the values of the new attributes (in this case *CurrentHue*, *CurrentSaturation*) from those of the old attributes (in this case *CurrentX* and *CurrentY*).

When moving from a mode to another mode that has a more restricted color range (e.g., *CurrentX*/*CurrentY* to *CurrentHue*/*CurrentSaturation*, or *CurrentHue*/*CurrentSaturation* to *ColorTemperatureMireds*) it is possible for the current color value to have no equivalent in the new mode. The behavior in such cases is manufacturer dependent, and therefore it is recommended to avoid color mode changes of this kind during usage.

5.2.2.3.3 Use of the OptionsMask & OptionsOverride fields[76]

The OptionsMask & OptionsOverride fields SHALL both be present or both omitted in the command. A temporary Options bitmap SHALL be created from the *Options* attribute, using the OptionsMask & OptionsOverride fields, if present. Each bit of the temporary Options bitmap SHALL be determined as follows:

Each bit in the *Options* attribute SHALL determine the corresponding bit in the temporary Options bitmap, unless the OptionsMask field is present and has the corresponding bit set to 1, in which case the corresponding bit in the OptionsOverride field SHALL determine the corresponding bit in the temporary Options bitmap.

The resulting temporary Options bitmap SHALL then be processed as defined in section 5.2.2.2.1.10.

5.2.2.3.4 Move to Hue Command

5.2.2.3.4.1 Payload Format

The Move to Hue command payload SHALL be formatted as illustrated in Figure 5-2.

Figure 5-2. Format of the Move to Hue Command Payload

Octets	1	1	2	0/1	0/1
Data Type	uint8	enum8	uint16	map8	map8
Field Name	Hue	Direction	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.4.2 Hue Field

The Hue field specifies the hue to be moved to.

5.2.2.3.4.3 Direction Field

The Direction field SHALL be one of the non-reserved values in Table 5-14.

Table 5-14. Values of the Direction Field

Fade Mode Value	**Description**
0x00	Shortest distance
0x01	Longest distance
0x02	Up
0x03	Down

5.2.2.3.4.4 Transition Time Field

The Transition Time field specifies, in 1/10ths of a second, the time that SHALL be taken to move to the new hue.

[76] CCB 2085

5.2.2.3.4.5 OptionsMask and OptionsOverride fields[77]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.4.6 Effect on Receipt

On receipt of this command, a device SHALL also set the *ColorMode* attribute to the value 0x00 and then SHALL move from its current hue to the value given in the Hue field.

The movement SHALL be continuous, i.e., not a step function, and the time taken to move to the new hue SHALL be equal to the Transition Time field.

As hue is effectively measured on a circle, the new hue MAY be moved to in either direction. The direction of hue change is given by the Direction field. If Direction is 'Shortest distance', the direction is taken that involves the shortest path round the circle. This case corresponds to expected normal usage. If Direction is 'Longest distance', the direction is taken that involves the longest path round the circle. This case can be used for 'rainbow effects'. In both cases, if both distances are the same, the Up direction SHALL be taken.

5.2.2.3.5 Move Hue Command

5.2.2.3.5.1 Payload Format

The Move Hue command payload SHALL be formatted as illustrated in Figure 5-3.

Figure 5-3. Format of the Move Hue Command Payload

Octets	1	1	0/1	0/1
Data Type	enum8	uint8	map8	map8
Field Name	Move Mode	Rate	OptionsMask	OptionsOverride

5.2.2.3.5.2 Move Mode Field

The Move Mode field SHALL be one of the non-reserved values in Table 5-15.

Table 5-15. Values of the Move Mode Field

Fade Mode Value	**Description**
0x00	Stop
0x01	Up
0x02	Reserved
0x03	Down

5.2.2.3.5.3 Rate Field

The Rate field specifies the rate of movement in steps per second. A step is a change in the device's hue of one unit. If the Rate field has a value of zero, the command has no effect and a default response command (see Chapter 2) is sent in response, with the status code set to INVALID_FIELD.

[77] CCB 2085

5.2.2.3.5.4 OptionsMask and OptionsOverride field[78]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.5.5 Effect on Receipt

On receipt of this command, a device SHALL set the *ColorMode* attribute to the value 0x00 and SHALL then move from its current hue in an up or down direction in a continuous fashion, as detailed in Table 5-16.

Table 5-16. Actions on Receipt for Move Hue Command

Fade 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 a Move to Hue command, a Move to Saturation command, or a Move to Hue and Saturation command.
Up	Increase the device's hue at the rate given in the Rate field. If the hue reaches the maximum allowed for the device, then proceed to its minimum allowed value.
Down	Decrease the device's hue at the rate given in the Rate field. If the hue reaches the minimum allowed for the device, then proceed to its maximum allowed value.

5.2.2.3.6 Step Hue Command

5.2.2.3.6.1 Payload Format

The Step Hue command payload SHALL be formatted as illustrated in Figure 5-4.

Figure 5-4. Format of the Step Hue Command Payload

Octets	1	1	1	0/1	0/1
Data Type	enum8	uint8	uint8	map8	map8
Field Name	Step Mode	Step Size	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.6.2 Step Mode Field

The Step Mode field SHALL be one of the non-reserved values in Table 5-17.

Table 5-17. Values of the Step Mode Field

Fade Mode Value	Description
0x00	Reserved
0x01	Up
0x02	Reserved

[78] CCB 2085

0x03	Down

5.2.2.3.6.3 Step Size Field

The change to be added to (or subtracted from) the current value of the device's hue.

5.2.2.3.6.4 Transition Time Field

The Transition Time field specifies, in 1/10ths of a second, the time that SHALL be taken to perform the step. A step is a change in the device's hue of 'Step size' units.

5.2.2.3.6.5 OptionsMask and OptionsOverride fields[79]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.6.6 Effect on Receipt

On receipt of this command, a device SHALL set the *ColorMode* attribute to the value 0x00 and SHALL then move from its current hue in an up or down direction by one step, as detailed in Table 5-18.

Table 5-18. Actions on Receipt for Step Hue Command

Fade Mode	Action on Receipt
Up	Increase the device's hue by one step, in a continuous fashion. If the hue value reaches the maximum value then proceed to the minimum allowed value.
Down	Decrease the device's hue by one step, in a continuous fashion. If the hue value reaches the minimum value then proceed to the maximum allowed value.

5.2.2.3.7 Move to Saturation Command

5.2.2.3.7.1 Payload Format

The Move to Saturation command payload SHALL be formatted as illustrated in Figure 5-5.

Figure 5-5. Format of the Move to Saturation Command Payload

Octets	1	2	0/1	0/1
Data Type	uint8	uint16	map8	map8
Field Name	Saturation	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.7.2 OptionsMask and OptionsOverride fields

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.7.3 Effect on Receipt

[79] CCB 2085

On receipt of this command, a device set the *ColorMode* attribute to the value 0x00 and SHALL then move from its current saturation to the value given in the Saturation field.

The movement SHALL be continuous, i.e., not a step function, and the time taken to move to the new saturation SHALL be equal to the Transition Time field, in 1/10ths of a second.

5.2.2.3.8 Move Saturation Command

5.2.2.3.8.1 Payload Format

The Move Saturation command payload SHALL be formatted as illustrated in Figure 5-6.

Figure 5-6. Format of the Move Saturation Command Payload

Octets	1	1	0/1	0/1
Data Type	enum8	uint8	map8	map8
Field Name	Move Mode	Rate	OptionsMask	OptionsOverride

5.2.2.3.8.2 Move Mode Field

The Move Mode field SHALL be one of the non-reserved values in Table 5-19.

Table 5-19. Values of the Move Mode Field

Fade Mode Value	**Description**
0x00	Stop
0x01	Up
0x02	Reserved
0x03	Down

5.2.2.3.8.3 Rate Field

The Rate field specifies the rate of movement in steps per second. A step is a change in the device's saturation of one unit. If the Rate field has a value of zero, the command has no effect and a default response command (see Chapter 2) is sent in response, with the status code set to INVALID_FIELD.

5.2.2.3.8.4 OptionsMask and OptionsOverride fields[80]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.8.5 Effect on Receipt

On receipt of this command, a device SHALL set the *ColorMode* attribute to the value 0x00 and SHALL then move from its current saturation in an up or down direction in a continuous fashion, as detailed in Table 5-20.

[80] CCB 2085

Table 5-20. Actions on Receipt for Move Saturation Command

Fade Mode	Action on Receipt
Stop	If moving, stop, else ignore the command (i.e., the command is accepted but has no affect). NB This MAY also be used to stop a Move to Saturation command, a Move to Hue command, or a Move to Hue and Saturation command.
Up	Increase the device's saturation at the rate given in the Rate field. If the saturation reaches the maximum allowed for the device, stop.
Down	Decrease the device's saturation at the rate given in the Rate field. If the saturation reaches the minimum allowed for the device, stop.

5.2.2.3.9 Step Saturation Command

5.2.2.3.9.1 Payload Format

The Step Saturation command payload SHALL be formatted as illustrated in Figure 5-7.

Figure 5-7. Format of the Step Saturation Command Payload

Octets	1	1	1	0/1	0/1
Data Type	enum8	uint8	uint8	map8	map8
Field Name	Step Mode	Step Size	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.9.2 Step Mode Field

The Step Mode field SHALL be one of the non-reserved values in Table 5-21.

Table 5-21. Values of the Step Mode Field

Step Mode Value	Description
0x00	Reserved
0x01	Up
0x02	Reserved
0x03	Down

5.2.2.3.9.3 Step Size Field

The change to be added to (or subtracted from) the current value of the device's saturation.

5.2.2.3.9.4 Transition Time Field

The Transition Time field specifies, in 1/10ths of a second, the time that SHALL be taken to perform the step. A step is a change in the device's saturation of 'Step size' units.

5.2.2.3.9.5 OptionsMask and OptionsOverride fields[81]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.9.6 Effect on Receipt

On receipt of this command, a device SHALL set the *ColorMode* attribute to the value 0x00 and SHALL then move from its current saturation in an up or down direction by one step, as detailed in Table 5-22.

Table 5-22. Actions on Receipt for Step Saturation Command

Step Mode	Action on Receipt
Up	Increase the device's saturation by one step, in a continuous fashion. However, if the saturation value is already the maximum value then do nothing.
Down	Decrease the device's saturation by one step, in a continuous fashion. However, if the saturation value is already the minimum value then do nothing.

5.2.2.3.10 Move to Hue and Saturation Command

5.2.2.3.10.1 Payload Format

The Move to Hue and Saturation command payload SHALL be formatted as illustrated in Figure 5-8.

Figure 5-8. Move to Hue and Saturation Command Payload

Octets	1	1	2	0/1	0/1
Data Type	uint8	uint8	uint16	map8	map8
Field Name	Hue	Saturation	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.10.2 OptionsMask and OptionsOverride fields[82]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.10.3 Effect on Receipt

On receipt of this command, a device SHALL set the *ColorMode* attribute to the value 0x00 and SHALL then move from its current hue and saturation to the values given in the 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 hue/saturation space, i.e., movement is 'in a straight line' across the hue/saturation disk.

[81] CCB 2085
[82] CCB 2085

5.2.2.3.11 Move to Color Command

5.2.2.3.11.1 Payload Format

The Move to Color command payload SHALL be formatted as illustrated in Figure 5-9.

Figure 5-9. Format of the Move to Color Command Payload

Octets	2	2	2	0/1	0/1
Data Type	uint16	uint16	uint16	map8	map8
Field Name	ColorX	ColorY	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.11.2 OptionsMask and OptionsOverride fields

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.11.3 Effect on Receipt

On receipt of this command, a device SHALL set the value of the *ColorMode* attribute, where implemented, to 0x01, and SHALL then move from its current color to the color given in the ColorX and ColorY 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 color space, i.e., movement is 'in a straight line' across the CIE xyY Color Space.

5.2.2.3.12 Move Color Command

5.2.2.3.12.1 Payload Format

The Move Color command payload SHALL be formatted as illustrated in Figure 5-10.

Figure 5-10. Format of the Move Color Command Payload

Octets	2	2	0/1	0/1
Data Type	int16	int16	map8	map8
Field Name	RateX	RateY	OptionsMask	OptionsOverride

5.2.2.3.12.2 RateX Field

The RateX field specifies the rate of movement in steps per second. A step is a change in the device's CurrentX attribute of one unit.

5.2.2.3.12.3 RateY Field

The RateY field specifies the rate of movement in steps per second. A step is a change in the device's CurrentY attribute of one unit.

5.2.2.3.12.4 OptionsMask and OptionsOverride fields[83]

[83] CCB 2085

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.12.5 Effect on Receipt

On receipt of this command, a device SHALL set the value of the *ColorMode* attribute, where implemented, to 0x01, and SHALL then move from its current color in a continuous fashion according to the rates specified. This movement SHALL continue until the target color for the next step cannot be implemented on this device.

If both the RateX and RateY fields contain a value of zero, no movement SHALL be carried out, and the command execution SHALL have no effect other than stopping the operation of any previously received command of this cluster. This command can thus be used to stop the operation of any other command of this cluster.

5.2.2.3.13 Step Color Command

5.2.2.3.13.1 Payload Format

The Step Color command payload SHALL be formatted as illustrated in Figure 5-11.

Figure 5-11. Format of the Step Color Command Payload

Octets	2	2	2	0/1	0/1
Data Type	int16	int16	uint16	map8	map8
Field Name	StepX	StepY	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.13.2 StepX and StepY Fields

The StepX and StepY fields specify the change to be added to the device's CurrentX attribute and CurrentY attribute respectively.

5.2.2.3.13.3 Transition Time Field

The Transition Time field specifies, in 1/10ths of a second, the time that SHALL be taken to perform the color change. 9999

5.2.2.3.13.4 OptionsMask and OptionsOverride fields[84]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.13.5 Effect on Receipt

On receipt of this command, a device SHALL set the value of the *ColorMode* attribute, where implemented, to 0x01, and SHALL then move from its current color by the color step indicated.

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 color space, i.e., movement is 'in a straight line' across the CIE xyY Color Space.

Note also that if the required step is larger than can be represented by signed 16-bit integers then more than one step command SHOULD be issued.

[84] CCB 2085

5.2.2.3.14 Move to Color Temperature Command

5.2.2.3.14.1 Payload Format

The Move to Color Temperature command payload SHALL be formatted as illustrated in Figure 5-12.

Figure 5-12. Move to Color Temperature Command Payload

Octets	2	2	0/1	0/1
Data Type	uint16	uint16	map8	map8
Field Name	Color Temperature Mireds	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.14.2 OptionsMask and OptionsOverride fields

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.14.3 Effect on Receipt

On receipt of this command, a device SHALL set the value of the *ColorMode* attribute, where implemented, to 0x02, and SHALL then move from its current color to the color given by the Color Temperature Mireds field.

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.

By definition of this color mode, the path through color space taken during the transition is along the 'Black Body Line'.

5.2.2.3.15 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 5-13.

Figure 5-13. Format of the Enhanced Move to Hue Command

Octets	2	1	2	0/1	0/1
Data Type	uint16	enum8	uint16	map8	map8
Field Name	Enhanced Hue	Direction	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.15.1 Enhanced Hue Field

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

5.2.2.3.15.2 Direction Field

This field is identical to the Direction field of the Move to Hue command of the Color Control cluster (see sub-clause 5.2.2.3.3).

5.2.2.3.15.3 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 5.2.2.3.3).

5.2.2.3.15.4 OptionsMask and OptionsOverride fields[85]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.15.5 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.

5.2.2.3.16 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 5-14.

Figure 5-14. Format of the Enhanced Move Hue Command

Octets	1	2	0/1	0/1
Data Type	enum8	uint16	map8	map8
Field Name	Move Mode	Rate	OptionsMask	OptionsOverride

5.2.2.3.16.1 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 5.2.2.3.5).

5.2.2.3.16.2 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 Move Mode field is set to 0x01 (up) or 0x03 (down) and 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. If the Move Mode field is set to 0x00 (stop) the rate field SHALL be ignored.

5.2.2.3.16.3 OptionsMask and OptionsOverride fields[86]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.16.4 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 in an up or down direction in a continuous fashion, as detailed in Table 5-23.

Table 5-23. Actions on Receipt of the Enhanced Move Hue Command

Move Mode	**Action on Receipt**

[85] CCB 2085
[86] CCB 2085

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.

5.2.2.3.17 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 5-15.

Figure 5-15. Format of the Enhanced Step Hue Command

Octets	1	2	2	0/1	0/1
Data Type	enum8	uint16	uint16	map8	map8
Field Name	Step Mode	Step Size	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.17.1 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 5.2.2.3.6).

5.2.2.3.17.2 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.

5.2.2.3.17.3 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.

5.2.2.3.17.4 OptionsMask and OptionsOverride fields[87]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.17.5 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 5-24.

[87] CCB 2085

Table 5-24. 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.

5.2.2.3.18 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 5-16.

Figure 5-16. Format of the Enhanced Move to Hue and Saturation Command

Octets	2	1	2	0/1	0/1
Data Type	uint16	uint8	uint16	map8	map8
Field Name	Enhanced Hue	Saturation	Transition Time	OptionsMask	OptionsOverride

5.2.2.3.18.1 Enhanced Hue Field

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

5.2.2.3.18.2 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 5.2.2.3.10).

5.2.2.3.18.3 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 5.2.2.3.10).

5.2.2.3.18.4 OptionsMask and OptionsOverride fields

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.18.5 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.

5.2.2.3.19 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 5-17.

Figure 5-17. Format of the Color Loop Set Command

Octets	1	1	1	2	2	0/1	0/1
Data Type	map8	enum8	enum8	uint16	uint16	map8	map8
Field Name	Update Flags	Action	Direction	Time	Start Hue	OptionsMask	OptionsOverride

5.2.2.3.19.1 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 5-18.

Figure 5-18. Format of the Update Flags Field of the Color Loop Set Command

Bits: 0	**1**	**2**	**3**	**4-7**
Update Action	Update Direction	Update Time	Update Start Hue	Reserved

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.

5.2.2.3.19.2 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 5-25.

Table 5-25. Values of the Action Field of the Color Loop Set Command

Value	**Description**
0x00	De-activate the color loop.

Value	Description
0x01	Activate the color loop from the value in the *ColorLoopStartEnhancedHue* field.
0x02	Activate the color loop from the value of the *EnhancedCurrentHue* attribute.

5.2.2.3.19.3 Direction Field

The 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 5-26.

Table 5-26. 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.

5.2.2.3.19.4 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.

5.2.2.3.19.5 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.

5.2.2.3.19.6 OptionsMask and OptionsOverride fields[88]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.19.7 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 and the color loop is active, i.e. the ColorLoopActive attribute is set to 0x01, 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 0x00 and the color loop is inactive, i.e. the ColorLoopActive attribute is set to 0x00, the device SHALL ignore the action update component of the command. 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.

[88] CCB 2085

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.

5.2.2.3.20 Stop Move Step Command

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.)

Note: the Stop Move Step command has no effect on an active color loop.

The Stop Move Step command payload SHALL be formatted as illustrated in Figure 5-19.

Figure 5-19. Format of the Stop Move Step Command Payload

Octets	0/1	0/1
Data Type	map8	map8
Field Name	OptionsMask	OptionsOverride

5.2.2.3.20.1 OptionsMask and OptionsOverride fields[89]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.20.2 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.

5.2.2.3.21 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 5-20.

Figure 5-20. Format of the Move Color Temperature Command

Octets	1	2	2	2	0/1	0/1
Data Type	map8	uint16	uint16	uint16	map8	map8
Field Name	Move Mode	Rate	Color Temperature Minimum Mireds	Color Temperature Maximum Mireds	OptionsMask	OptionsOverride

5.2.2.3.21.1 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 5.2.2.3.5).

5.2.2.3.21.2 Rate Field

[89] CCB 2085

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.

5.2.2.3.21.3 Color Temperature Minimum Mireds Field

The Color Temperature Minimum Mireds field is 16-bits in length and specifies a lower bound on the *ColorTemperatureMireds* attribute (≡ an upper bound on the color temperature in kelvins) for the current move operation such that:

ColorTempPhysicalMinMireds ≤ Color Temperature Minimum Mireds field ≤ ColorTemperatureMireds

As such if the move operation takes the ColorTemperatureMireds attribute towards the value of the Color Temperature Minimum Mireds field it SHALL be clipped so that the above invariant is satisfied. If the Color Temperature Minimum Mireds field is set to 0x0000, ColorTempPhysicalMinMireds SHALL be used as the lower bound for the ColorTemperatureMireds attribute.

5.2.2.3.21.4 Color Temperature Maximum Mireds Field

The Color Temperature Maximum Mireds field is 16-bits in length and specifies an upper bound on the *ColorTemperatureMireds* attribute (≡ a lower bound on the color temperature in kelvins) for the current move operation such that:

ColorTemperatureMireds ≤ Color Temperature Maximum Mireds field ≤ ColorTempPhysicalMaxMireds

As such if the move operation takes the ColorTemperatureMireds attribute towards the value of the Color Temperature Maximum Mireds field it SHALL be clipped so that the above invariant is satisfied. If the Color Temperature Maximum Mireds field is set to 0x0000, ColorTempPhysicalMaxMireds SHALL be used as the upper bound for the ColorTemperatureMireds attribute.

5.2.2.3.21.5 OptionsMask and OptionsOverride fields[90]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.21.6 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 in a continuous fashion, as detailed in Table 5-27.

Table 5-27. 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 *ColorTemperatureMireds* attribute (≡ decrease the color temperature in kelvins) at the rate given in the Rate field. If the *ColorTemperatureMireds* attribute reaches the maximum allowed for the device (via either the Color Temperature Maximum Mireds field or the *ColorTempPhysicalMaxMireds* attribute), the move operation SHALL be stopped.
Down	Decrease the *ColorTemperatureMireds* attribute (≡ increase the color temperature in kelvins) at the rate given in the Rate field. If the *ColorTemperatureMireds* attribute reaches the minimum allowed for the device (via either the Color Temperature Minimum Mireds field or the *ColorTempPhysicalMinMireds* attribute), the move operation SHALL be stopped.

[90] CCB 2085

5.2.2.3.22 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 5-21.

Figure 5-21. Format of the Step Color Temperature Command

Octets	1	2	2	2	2	0/1	0/1
Data Type	map8	uint16	uint16	uint16	uint16	map8	map8
Field Name	Step Mode	Step Size	Transition Time	Color Temperature Minimum Mireds	Color Temperature Maximum Mireds	Options Mask	Options Override

5.2.2.3.22.1 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 5.2.2.3.6).

5.2.2.3.22.2 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.

5.2.2.3.22.3 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.

5.2.2.3.22.4 Color Temperature Minimum Mireds Field

The Color Temperature Minimum Mireds field is 16-bits in length and specifies a lower bound on the *ColorTemperatureMireds* attribute (≡ an upper bound on the color temperature in kelvins) for the current step operation such that:

ColorTempPhysicalMinMireds ≤ Color Temperature Minimum Mireds field ≤ ColorTemperatureMireds

As such if the step operation takes the ColorTemperatureMireds attribute towards the value of the Color Temperature Minimum Mireds field it SHALL be clipped so that the above invariant is satisfied. If the Color Temperature Minimum Mireds field is set to 0x0000, ColorTempPhysicalMinMireds SHALL be used as the lower bound for the ColorTemperatureMireds attribute.

5.2.2.3.22.5 Color Temperature Maximum Mireds Field

The Color Temperature Maximum Mireds field is 16-bits in length and specifies an upper bound on the *ColorTemperatureMireds* attribute (≡ a lower bound on the color temperature in kelvins) for the current step operation such that:

ColorTemperatureMireds ≤ Color Temperature Maximum Mireds field ≤ *ColorTempPhysicalMaxMireds*

As such if the step operation takes the *ColorTemperatureMireds* attribute towards the value of the Color Temperature Maximum Mireds field it SHALL be clipped so that the above invariant is satisfied. If the Color Temperature Maximum Mireds field is set to 0x0000, *ColorTempPhysicalMaxMireds* SHALL be used as the upper bound for the *ColorTemperatureMireds* attribute.

5.2.2.3.22.6 OptionsMask and OptionsOverride fields[91]

The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

5.2.2.3.22.7 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 5-28.

Table 5-28. Actions on Receipt of the Step Color Temperature Command

Move Mode	Action on Receipt
Up	Increase the *ColorTemperatureMireds* attribute (≡ decrease the color temperature in kelvins) by one step. If the *ColorTemperatureMireds* attribute reaches the maximum allowed for the device (via either the Color Temperature Maximum Mireds field or the *ColorTempPhysicalMaxMireds* attribute), the step operation SHALL be stopped.
Down	Decrease the *ColorTemperatureMireds* attribute (≡ increase the color temperature in kelvins) by one step. If the *ColorTemperatureMireds* attribute reaches the minimum allowed for the device (via either the Color Temperature Minimum Mireds field or the *ColorTempPhysicalMinMireds* attribute), the step operation SHALL be stopped.

5.2.2.4 Commands Generated

The server generates no cluster specific commands

5.2.2.5 Scene Table Extensions

If the Scenes server cluster (see 3.7) is implemented, the following extension fields SHALL be added to the Scenes table in the given order, i.e., the attribute listed as 1 is added first:

1. *CurrentX*
2. *CurrentY*
3. *EnhancedCurrentHue*
4. *CurrentSaturation*
5. *ColorLoopActive*
6. *ColorLoopDirection*
7. *ColorLoopTime*
8. *ColorTemperatureMireds*

Since there is a direct relation between *ColorTemperatureMireds* 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.

[91] CCB 2085

5.2.2.6 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes command and according to the minimum and maximum reporting interval and reportable change settings described in the Chapter 2, Foundation. The following attributes SHALL be reportable:

CurrentX, CurrentY

CurrentHue (if implemented), *CurrentSaturation* (if implemented), *ColorTemperatureMireds* (if implemented)

5.2.3 Client

The client has no specific dependencies, no specific attributes, and receives no cluster specific commands. The client generates the cluster specific commands detailed in 5.2.2.3, as required by the application.

5.3 Ballast Configuration Cluster

5.3.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster is used for configuring a lighting ballast.

5.3.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added
2	CCB 2104 2193 2230 2393 Deprecated some attributes

5.3.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BALCFG	Type 1 (client to server)

5.3.1.3 Cluster Identifiers

Identifier	Name
0x0301	Ballast Configuration

5.3.2 Server

5.3.2.1 Dependencies

For the alarm functionality specified by this cluster to be operational, the Alarms server cluster SHALL be implemented on the same endpoint.

5.3.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 5-29.

Table 5-29. Ballast Configuration Attribute Sets

Attribute Set Identifier	Description
0x000	Ballast information
0x001	Ballast settings
0x002	Lamp information
0x003	Lamp settings

5.3.2.2.1 Ballast Information Attribute Set

The Ballast Information attribute set contains the attributes summarized in Table 5-30.

Table 5-30. Attributes of the Ballast Information Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0000	PhysicalMinLevel	uint8	0x01 – 0xfe	R	0x01	M[92]
0x0001	PhysicalMaxLevel	uint8	0x01 – 0xfe	R	0xfe	M[93]
0x0002	BallastStatus	map8	0000 00xx	R	0000 0000	O[94]

5.3.2.2.1.1 *PhysicalMinLevel* Attribute

The *PhysicalMinLevel* attribute is 8 bits in length and specifies the minimum light output[95] the ballast can achieve. This attribute SHALL be specified in the range 0x01 to 0xfe, and specifies the light output of the ballast according to the dimming light curve (see 5.3.4).

5.3.2.2.1.2 *PhysicalMaxLevel* Attribute

The *PhysicalMaxLevel* attribute is 8 bits in length and specifies the maximum light output[96] the ballast can achieve according to the dimming light curve (see 5.3.4).

5.3.2.2.1.3 *BallastStatus* Attribute

[92] CCB 2193
[93] CCB 2193
[94] CCB 2193
[95] CCB 2104
[96] CCB 2104

The *BallastStatus* attribute is 8 bits in length and specifies the activity status of the ballast functions. The usage of the bits is specified in Table 5-31. Where a function is active, the corresponding bit SHALL be set to 1. Where a function is not active, the corresponding bit SHALL be set to 0.

Table 5-31. Bit Usage of the *BallastStatus* Attribute

Bit	Function	Details
0	Ballast Non-operational	0 = The ballast is fully operational 1 = The ballast is not fully operational
1	Lamp Failure[97]	0 = All lamps are operational 1 = One or more lamp is not in its socket or is faulty

5.3.2.2.2 Ballast Settings Attribute Set

The Ballast Settings attribute set contains the attributes summarized in Table 5-32.

Table 5-32. Attributes of the Ballast Settings Attribute Set

Id	Name	Type	Range	Acc	Default	M/O
0x0010	MinLevel	uint8	0x01 – 0xfe	RW	*PhysicalMinLevel*	M
0x0011	MaxLevel	uint8	0x01 – 0xfe	RW	*PhysicalMaxLevel*	M
0x0012	PowerOnLevel	uint8	0x00 – 0xfe	RW	*PhysicalMaxLevel*	D[98]
0x0013	PowerOnFadeTime	uint16	0x0000 – 0xfffe	RW	0x0000	D[99]
0x0014	IntrinsicBallastFactor	uint8	0x00 – 0xfe	RW	-	O
0x0015	BallastFactorAdjustment	uint8	0x64 – MS	RW	0xff	O

5.3.2.2.2.1 *MinLevel* Attribute

The *MinLevel* attribute is 8 bits in length and specifies the light output of the ballast according to the dimming light curve (see 5.3.4) when the Level Control Cluster's CurrentLevel attribute equals to 0x01 (1) (and the On/Off Clusters's OnOff attribute equals to 0x01)[100].

The value of this attribute SHALL be both greater than or equal to *PhysicalMinLevel* and less than or equal to *MaxLevel*. If an attempt is made to set this attribute to a level where these conditions are not met, a default response command SHALL be returned with status code set to INVALID_VALUE, and the level SHALL not be set.

5.3.2.2.2.2 *MaxLevel* Attribute

The *MaxLevel* attribute is 8 bits in length and specifies the light output of the ballast according to the dimming light curve (see 5.3.4) when the Level Control Cluster's CurrentLevel attribute equals to 0xfe (254) (and the On/Off Cluster's OnOff attribute equals to 0x01)[101].

[97] CCB 2230 Updated to align with DALI
[98] CCB 2393 Deprecated: see Chapter 2 for more information
[99] CCB 2393 Deprecated: see Chapter 2 for more information
[100] CCB 2104
[101] CCB 2104

The value of this attribute SHALL be both less than or equal to *PhysicalMaxLevel* and greater than or equal to *MinLevel*. If an attempt is made to set this attribute to a level where these conditions are not met, a default response command SHALL be returned with status code set to INVALID_VALUE, and the level SHALL not be set.

5.3.2.2.2.3 *IntrinsicBallastFactor* Attribute

The *IntrinsicBallastFactor* attribute is 8 bits in length and specifies as a percentage the ballast factor of the ballast/lamp combination (see also 5.3), prior to any adjustment.

A value of 0xff indicates in invalid value.

5.3.2.2.2.4 *BallastFactorAdjustment* Attribute

The *BallastFactorAdjustment* attribute is 8 bits in length and specifies the multiplication factor, as a percentage, to be applied to the configured light output of the lamps (see also Overview5.3). A typical usage of this mechanism is to compensate for reduction in efficiency over the lifetime of a lamp.

The light output is given by

Actual light output = configured light output x *BallastFactorAdjustment* / 100%

The range for this attribute is manufacturer dependent. If an attempt is made to set this attribute to a level that cannot be supported, a default response command SHALL be returned with status code set to INVALID_VALUE, and the level SHALL not be set. The value 0xff indicates that ballast factor scaling is not in use.

5.3.2.2.3 Lamp Information Attribute Set

The lamp information attribute set contains the attributes summarized in Table 5-33.

Table 5-33. Attributes of the Lamp Information Attribute Set

Identifier	Name	Type	Range	Access	Default	M/O
0x0020	LampQuantity	uint8	0x00 – 0xfe	R	-	O

5.3.2.2.3.1 *LampQuantity* Attribute

The *LampQuantity* attribute is 8 bits in length and specifies the number of lamps connected to this ballast. (**Note 1:** this number does not take into account whether lamps are actually in their sockets or not).

5.3.2.2.4 Lamp Settings Attribute Set

The Lamp Settings attribute set contains the attributes summarized in Table 5-34. If *LampQuantity* is greater than one, each of these attributes is taken to apply to the lamps as a set. For example, all lamps are taken to be of the same *LampType* with the same *LampBurnHours.*

Table 5-34. Attributes of the Lamp Settings Attribute Set

Id	Name	Type	Range	Acc	Default	M/O
0x0030	LampType	string	-	RW	empty string	O
0x0031	LampManufacturer	string	-	RW	empty string	O
0x0032	LampRatedHours	uint24	0x000000 – 0xfffffe	RW	0xffffff	O
0x0033	LampBurnHours	uint24	0x000000 – 0xfffffe	RW	0x000000	O

0x0034	LampAlarmMode	map8	0000 000x	RW	0000 0000	O
0x0035	LampBurnHoursTripPoint	uint24	0x000000 – 0xfffffe	RW	0xffffff	O

5.3.2.2.4.1 *LampType* Attribute

The *LampType* attribute is a character string of up to 16 bytes in length. It specifies the type of lamps (including their wattage) connected to the ballast.

5.3.2.2.4.2 *LampManufacturer* Attribute

The *LampManufacturer* attribute is a character string of up to 16 bytes in length. It specifies the name of the manufacturer of the currently connected lamps.

5.3.2.2.4.3 *LampRatedHours* Attribute

The *LampRatedHours* attribute is 24 bits in length and specifies the number of hours of use the lamps are rated for by the manufacturer.

A value of 0xffffff indicates an invalid or unknown time.

5.3.2.2.4.4 *LampBurnHours* Attribute

The *LampBurnHours* attribute is 24 bits in length and specifies the length of time, in hours, the currently connected lamps have been operated, cumulative since the last re-lamping. Burn hours SHALL not be accumulated if the lamps are off.

This attribute SHOULD be reset to zero (e.g., remotely) when the lamp(s) are changed. If partially used lamps are connected, *LampBurnHours* SHOULD be updated to reflect the burn hours of the lamps.

A value of 0xffffff indicates an invalid or unknown time.

5.3.2.2.4.5 *LampAlarmMode* Attribute

The *LampsAlarmMode* attribute is 8 bits in length and specifies which attributes MAY cause an alarm notification to be generated, as listed in Table 5-35. A '1' in each bit position causes its associated attribute to be able to generate an alarm. (**Note:** All alarms are also logged in the alarm table – see Alarms cluster 3.11).

Table 5-35. Values of the *MainsAlarmMode* Attribute

Attribute Bit Number	Attribute
0	LampBurnHours

5.3.2.2.4.6 *LampBurnHoursTripPoint* Attribute

The *LampBurnHoursTripPoint* attribute is 24 bits in length and specifies the number of hours the LampBurnHours attribute MAY reach before an alarm is generated.

If the Alarms cluster is not present on the same device this attribute is not used and thus MAY be omitted (see 5.3.2.1).

The Alarm Code field included in the generated alarm SHALL be 0x01.

If this attribute takes the value 0xffffff then this alarm SHALL not be generated.

5.3.2.3 Commands

No cluster specific commands are received or generated by the server.

5.3.3 Client

The client has no attributes. No cluster specific commands are received by the client. No cluster specific commands are generated by the client.

5.3.4 The Dimming Light Curve

The dimming curve is recommended to be logarithmic, as defined by the following equation:

$$\%Light = 10^{\left(\frac{Level-1}{\left(\frac{253}{3}\right)}\right)-1}$$

Where: %Light is the percent light output of the ballast and

Level is an 8-bit integer between 1 (0.1% light output) and 254 (100% output) that is adjusted for MinLevel and MaxLevel using the following equation[102]:

Level = (MaxLevel – MinLevel) * CurrentLevel / 253 + (254 * MinLevel – MaxLevel) / 253.

Note 1: Value 255 is not used.

Note 2: The light output is determined by this curve together with the *IntrinsicBallastFactor* and *BallastFactorAdjustment* Attributes.

The table below gives a couple of examples of the dimming light curve for different values of MinLevel, MaxLevel and CurrentLevel[103].

Table 5-36. Examples of The Dimming Light Curve

MinLevel	MaxLevel	CurrentLevel	Level	%Light
1	254	1	1	0.10%
1	254	10	10	0.13%
1	254	100	100	1.49%
1	254	154	254	100%
170	254	1	170	10.1%
170	254	10	173	11.0%
170	254	100	203	24.8%
170	254	254	254	100%
170	230	1	170	10.1%
170	230	10	172	10.7%

102 CCB 2104
103 CCB 2104

170	230	100	193	19.2%
170	230	254	230	51.9%

CHAPTER 6 HVAC

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

6.1 General Description

6.1.1 Introduction

The clusters specified in this document are for use typically in HVAC applications, but MAY be used in any application domain.

6.1.2 Terms

4-pipes: In a 4-pipe HVAC fan coil system, heated and chilled water each have their own supply and return pipes, while in a 2 pipe system they share the same supply and return. With a 4-pipes system, heating and cooling can take place at the same time in different locations of a building. With a 2-pipes system, only heating or cooling can take place in the whole building.

Precooling: Cooling a building in the early (cooler) part of the day, so that the thermal mass of the building decreases cooling needs in the later (hotter) part of the day.

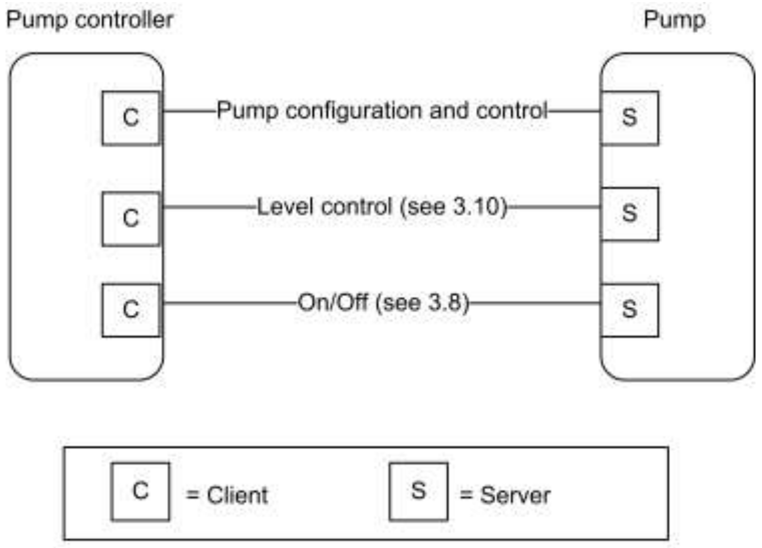
6.1.3 Cluster List

This section lists the clusters specified in this document, and gives examples of typical usage for the purpose of clarification. The clusters defined in this document are listed in Table 6-1:

Table 6-1. Clusters Specified in the HVAC Functional Domain

ID	Cluster Name	Description
0x0200	Pump Configuration and Control	An interface for configuring and controlling pumps.
0x0201	Thermostat	An interface for configuring and controlling the functionality of a thermostat
0x0202	Fan Control	An interface for controlling a fan in a heating / cooling system
0x0203	Dehumidification Control	An interface for controlling dehumidification
0x0204	Thermostat User Interface Configuration	An interface for configuring the user interface of a thermostat (which MAY be remote from the thermostat)

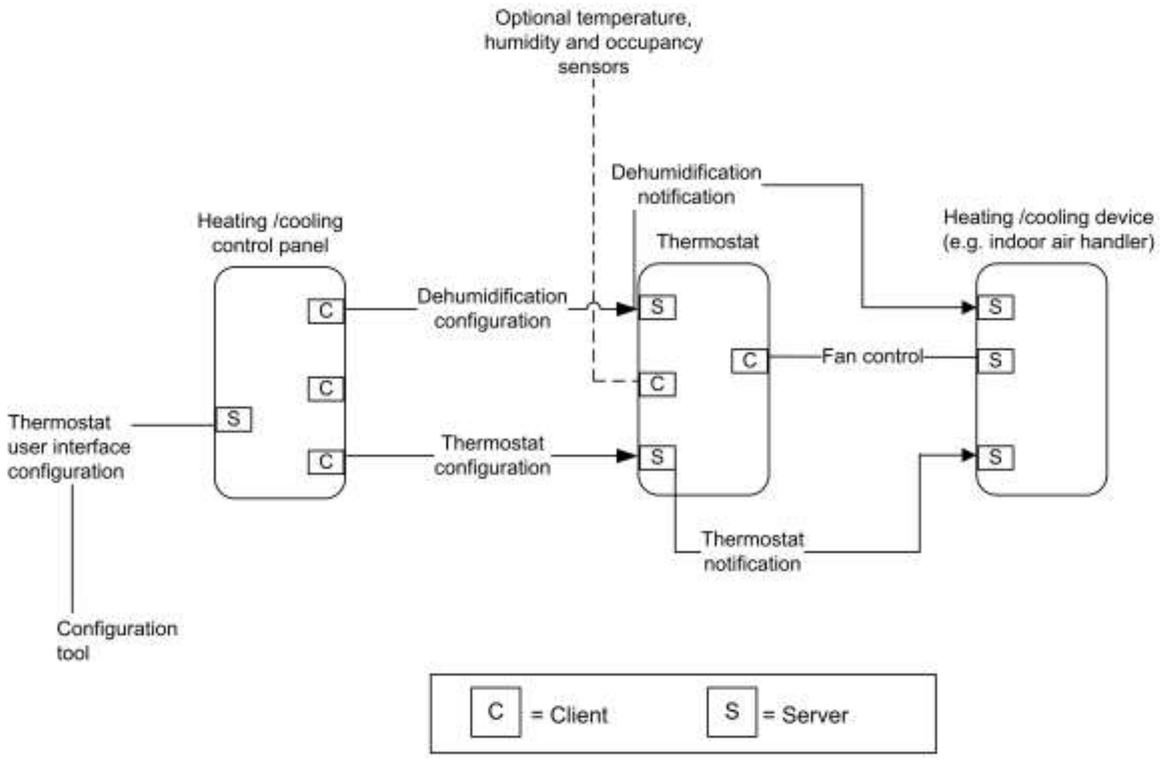
8813 **Figure 6-1. Typical Usage of Pump Configuration and Control Cluster**



8814 *Note: Device names are examples for illustration purposes only*

8815

8816 **Figure 6-2. Example Usage of the Thermostat and Related Clusters**


Optional temperature, humidity and occupancy sensors
Dehumidification notification
Heating /cooling control panel
Thermostat
Heating /cooling device (e.g. indoor air handler)
Dehumidification configuration
Fan control
Thermostat user interface configuration
Thermostat configuration
Thermostat notification
Configuration tool
C = Client
S = Server

8817 *Note: Device names are examples for illustration purposes only*

6.2 Pump Configuration and Control

6.2.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The Pump Configuration and Control cluster provides an interface for the setup and control of pump devices, and the automatic reporting of pump status information. Note that control of pump speed is not included – speed is controlled by the On/Off and Level Control clusters.

6.2.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

6.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	PCC	Type 2 (server to client)

6.2.1.3 Cluster Identifiers

Identifier	Name
0x0200	Pump Configuration and Control

6.2.2 Server

6.2.2.1 Dependencies

Where external pressure, flow, and temperature measurements are processed by this cluster (see Table 6-8), these are provided by a Pressure Measurement cluster (4.5), a Flow Measurement cluster (4.6), and a Temperature Measurement client cluster (4.4), respectively. These 3 client clusters are used for connection to a remote sensor device. The pump is able to use the sensor measurement provided by a remote sensor for regulation of the pump speed.

For the alarms, described in Table 6-9, to be operational, the Alarms server cluster (3.11) SHALL be implemented on the same endpoint.

Note that control of the pump setpoint is not included in this cluster – the On/Off and Level Control clusters (see Figure 6-1) MAY be used by a pump device to turn it on and off and control its setpoint. Note that the Pump Configuration and Control Cluster MAY override on/off/setpoint settings for specific operation modes (See section 6.2.2.2.3.1 for detailed description of the operation and control of the pump.).

6.2.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 6-2.

Table 6-2. Pump Configuration Attribute Sets

Attribute Set Identifier	Description
0x000	Pump Information
0x001	Pump Dynamic Information
0x002	Pump Settings

6.2.2.2.1 Pump Information Attribute Set

The pump information attribute set contains the attributes summarized in Table 6-3:

Table 6-3. Attributes of the Pump Information Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0000	*MaxPressure*	int16	0x8001-0x7fff	R	-	M
0x0001	*MaxSpeed*	uint16	0x0000 – 0xfffe	R	-	M
0x0002	*MaxFlow*	uint16	0x0000 – 0xfffe	R	-	M
0x0003	*MinConstPressure*	int16	0x8001-0x7fff	R	-	O
0x0004	*MaxConstPressure*	int16	0x8001-0x7fff	R	-	O
0x0005	*MinCompPressure*	int16	0x8001-0x7fff	R	-	O
0x0006	*MaxCompPressure*	int16	0x8001-0x7fff	R	-	O
0x0007	*MinConstSpeed*	uint16	0x0000 – 0xfffe	R	-	O
0x0008	*MaxConstSpeed*	uint16	0x0000 – 0xfffe	R	-	O
0x0009	*MinConstFlow*	uint16	0x0000 – 0xfffe	R	-	O
0x000a	*MaxConstFlow*	uint16	0x0000 – 0xfffe	R	-	O
0x000b	*MinConstTemp*	int16	0x954d – 0x7fff	R	-	O
0x000c	*MaxConstTemp*	int16	0x954d – 0x7fff	R	-	O

6.2.2.2.1.1 *MaxPressure* Attribute

The *MaxPressure* attribute specifies the maximum pressure the pump can achieve. It is a physical limit, and does not apply to any specific control mode or operation mode.

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is -3,276.7 kPa to 3,276.7 kPa (steps of 0.1 kPa). The value -3,276.8 kPa (0x8000) indicates that this value is invalid.

6.2.2.2.1.2 *MaxSpeed* Attribute

The *MaxSpeed* attribute specifies the maximum speed the pump can achieve. It is a physical limit, and does not apply to any specific control mode or operation mode.

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is 0 to 65,534 RPM (steps of 1 RPM). The value 65,535 RPM (0xffff) indicates that this value is invalid.

6.2.2.2.1.3 *MaxFlow* Attribute

The *MaxFlow* attribute specifies the maximum flow the pump can achieve. It is a physical limit, and does not apply to any specific control mode or operation mode.

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is 0 m^3/h to 6,553.4 m^3/h (steps of 0.1 m^3/h). The value 6,553.5 m^3/h (0xffff) indicates that this value is invalid.

6.2.2.2.1.4 *MinConstPressure* Attribute

The *MinConstPressure* attribute specifies the minimum pressure the pump can achieve when it is running and working in control mode constant pressure (*ControlMode* attribute of the Pump settings attribute set is set to Constant pressure).

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is –3,276.7 kPa to 3,276.7 kPa (steps of 0.1 kPa). The value -3,276.8 kPa (0x8000) indicates that this value is invalid.

6.2.2.2.1.5 *MaxConstPressure* Attribute

The *MaxConstPressure* attribute specifies the maximum pressure the pump can achieve when it is working in control mode constant pressure (*ControlMode* attribute of the Pump settings attribute set is set to Constant pressure).

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is –3,276.7 kPa to 3,276.7 kPa (steps of 0.1 kPa). The value -3,276.8 kPa (0x8000) indicates that this value is invalid.

6.2.2.2.1.6 *MinCompPressure* Attribute

The *MinCompPressure* attribute specifies the minimum compensated pressure the pump can achieve when it is running and working in control mode Proportional pressure (*ControlMode* attribute of the Pump settings attribute set is set to Proportional pressure).

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is –3,276.7 kPa to 3,276.7 kPa (steps of 0.1 kPa). The value -3,276.8 kPa (0x8000) indicates that this value is invalid.

6.2.2.2.1.7 *MaxCompPressure* Attribute

The *MaxCompPressure* attribute specifies the maximum compensated pressure the pump can achieve when it is working in control mode Proportional pressure (*ControlMode* attribute of the Pump settings attribute set is set to Proportional pressure).

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is –3,276.7 kPa to 3,276.7 kPa (steps of 0.1 kPa). The value -3,276.8 kPa (0x8000) indicates that this value is invalid.

6.2.2.2.1.8 *MinConstSpeed* Attribute

The *MinConstSpeed* attribute specifies the minimum speed the pump can achieve when it is running and working in control mode Constant speed (*ControlMode* attribute of the Pump settings attribute set is set to Constant speed).

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is 0 to 65,534 RPM (steps of 1 RPM). The value 65,535 RPM (0xffff) indicates that this value is invalid.

6.2.2.2.1.9 *MaxConstSpeed* Attribute

The *MaxConstSpeed* attribute specifies the maximum speed the pump can achieve when it is working in control mode Constant speed (*ControlMode* attribute of the Pump settings attribute set is set to Constant speed).

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is 0 to 65,534 RPM (steps of 1 RPM). The value 65,535 RPM (0xffff) indicates that this value is invalid.

6.2.2.2.1.10 *MinConstFlow* Attribute

The *MinConstFlow* attribute specifies the minimum flow the pump can achieve when it is running and working in control mode Constant flow (*ControlMode* attribute of the Pump settings attribute set is set to Constant flow).

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is 0 m^3/h to 6,553.4 m^3/h (steps of 0.1 m^3/h). The value 6,553.5 m^3/h (0xffff) indicates that this value is invalid.

6.2.2.2.1.11 *MaxConstFlow* Attribute

The *MaxConstFlow* attribute specifies the maximum flow the pump can achieve when it is running and working in control mode Constant flow (*ControlMode* attribute of the Pump settings attribute set is set to Constant flow).

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is 0 m^3/h to 6,553.4 m^3/h (steps of 0.1 m^3/h). The value 6,553.5 m^3/h (0xffff) indicates that this value is invalid.

6.2.2.2.1.12 *MinConstTemp* Attribute

The *MinConstTemp* attribute specifies the minimum temperature the pump can maintain in the system when it is running and working in control mode Constant temperature (*ControlMode* attribute of the Pump settings attribute set is set to Constant temperature).

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value.

Valid range is –273.15 °C to 327.67 °C (steps of 0.01 °C). The value -327.68°C (0x8000) indicates that this value is invalid.

6.2.2.2.1.13 *MaxConstTemp* Attribute

The *MaxConstTemp* attribute specifies the maximum temperature the pump can maintain in the system when it is running and working in control mode Constant temperature (*ControlMode* attribute of the Pump settings attribute set is set to Constant temperature).

This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will display the invalid value. *MaxConstTemp* SHALL be greater than or equal to *MinConstTemp*

Valid range is –273.15 °C to 327.67 °C (steps of 0.01 °C). The value -327.68°C (0x8000) indicates that this value is invalid.

6.2.2.2.2 Pump Dynamic Information Attribute Set

The pump dynamic information attribute set contains the attributes summarized in Table 6-4:

Table 6-4. Attributes of the Pump Dynamic Information Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0010	*PumpStatus*	map16	-	RP	-	O
0x0011	*EffectiveOperationMode*	enum8	0x00 – 0xfe	R	-	M
0x0012	*EffectiveControlMode*	enum8	0x00 – 0xfe	R	-	M
0x0013	*Capacity*	int16	0x0000-0x7fff	RP	-	M
0x0014	*Speed*	uint16	0x0000 - 0xfffe	R	-	O
0x0015	*LifetimeRunningHours*	uint24	0x000000 - 0xfffffe	RW	0	O
0x0016	*Power*	uint24	0x000000 - 0xfffffe	RW	-	O
0x0017	*LifetimeEnergyConsumed*	uint32	0x00000000 - 0xfffffffe	R	0	O

6.2.2.2.2.1 *PumpStatus* Attribute

The *PumpStatus* attribute specifies the activity status of the pump functions listed in Table 6-5. Where a pump controller function is active, the corresponding bit SHALL be set to 1. Where a pump controller function is not active, the corresponding bit SHALL be set to 0.

Table 6-5. Values of the *PumpStatus* Attribute

Bit	Function	Remarks
0	Device fault	A fault related to the pump device is detected (Corresponds to a Alarm code in the range 6-13, see Table 6-9)
1	Supply fault	A fault related to the supply to the pump is detected (Corresponds to a Alarm code in the range 0-5 or 13, see Table 6-9)
2	Speed low	Setpoint is too low to achieve
3	Speed high	Setpoint is too high to achieve
4	Local override	The pump is overridden by local control
5	Running	Pump is currently running
6	Remote Pressure	A remote pressure sensor is used as the sensor for the regulation of the pump. *EffectiveControlMode* is Constant pressure, and the setpoint for the pump is interpreted as a percentage of the range of the remote sensor ([*MinMeasuredValue* – *MaxMeasuredValue*])

Bit	Function	Remarks
7	Remote Flow	A remote flow sensor is used as the sensor for the regulation of the pump. *EffectiveControlModeI* is Constant flow, and the setpoint for the pump is interpreted as a percentage of the range of the remote sensor ([*MinMeasuredValue* – *MaxMeasuredValue*])
8	Remote Temperature	A remote temperature sensor is used as the sensor for the regulation of the pump. *EffectiveControlModeI* is Constant temperature, and setpoint is interpreted as a percentage of the range of the remote sensor ([*MinMeasuredValue* – *MaxMeasuredValue*])

6.2.2.2.2.2 *EffectiveOperationMode* Attribute

The *EffectiveOperationMode* attribute specifies current effective operation mode of the pump. The value of the *EffectiveOperationMode* attribute is the same as the *OperationMode* attribute of the Pump settings attribute set, except when it is overridden locally. See section 6.2.2.2.3.1 for a detailed description of the operation and control of the pump.

This attribute is read only.

Valid range is defined by the operation modes listed in Table 6-1.

6.2.2.2.2.3 *EffectiveControlMode* Attribute

The *EffectiveControlMode* attribute specifies the current effective control mode of the pump.

The *EffectiveControlMode* attribute contains the control mode that currently applies to the pump. It will have the value of the *ControlMode* attribute, unless a remote sensor is used as the sensor for regulation of the pump. In this case, *EffectiveControlMode* will display Constant pressure, Constant flow or Constant temperature if the remote sensor is a pressure sensor, a flow sensor or a temperature sensor respectively, regardless of the value of the *ControlMode* attribute.

See section 6.2.2.2.3.1 for detailed description of the operation and control of the pump. This attribute is read only.

Valid range is defined by the control modes listed in Table 6-8.

6.2.2.2.2.4 *Capacity* Attribute

The *Capacity* attribute specifies the actual capacity of the pump as a percentage of the effective maximum setpoint value. It is updated dynamically as the speed of the pump changes.

This attribute is read only. If the value is not available (the measurement or estimation of the speed is done in the pump), this attribute will contain the invalid value.

Valid range is 0 % to 163.835% (0.005 % granularity). Although the *Capacity* attribute is a signed value, values of capacity less than zero have no physical meaning.
The value -163.840 % (0x8000) indicates that this value is invalid.

6.2.2.2.2.5 *Speed* Attribute

The *Speed* attribute specifies the actual speed of the pump measured in RPM. It is updated dynamically as the speed of the pump changes.

This attribute is read only. If the value is not available (the measurement or estimation of the speed is done in the pump), this attribute will contain the invalid value.

Valid range is 0 to 65.534 RPM.
The value 65.535 RPM (0xffff) indicates that this value is invalid.

6.2.2.2.2.6 *LifetimeRunningHours* Attribute

The *LifetimeRunningHours* attribute specifies the accumulated number of hours that the pump has been powered and the motor has been running. It is updated dynamically as it increases. It is preserved over power cycles of the pump. if *LifeTimeRunningHours* rises above maximum value it "rolls over" and starts at 0 (zero).

This attribute is writeable, in order to allow setting to an appropriate value after maintenance. If the value is not available, this attribute will contain the invalid value.

Valid range is 0 to 16,777,214 hrs. The value 16,777,215 (0xffffff) indicates that this value is unknown.

6.2.2.2.2.7 *Power* Attribute

The *Power* attribute specifies the actual power consumption of the pump in Watts. The value of the *Power* attribute is updated dynamically as the power consumption of the pump changes.

This attribute is read only. If the value is not available (the measurement of power consumption is not done in the pump), this attribute will display the invalid value.

Valid range is 0 to 16,777,214 Watts. The value 16,777,215 (0xffffff) indicates that this value is unknown.

6.2.2.2.2.8 *LifetimeEnergyConsumed* Attribute

The *LifetimeEnergyConsumed* attribute specifies the accumulated energy consumption of the pump through the entire lifetime of the pump in kWh. The value of the *LifetimeEnergyConsumed* attribute is updated dynamically as the energy consumption of the pump increases. If *LifetimeEnergyConsumed* rises above maximum value it "rolls over" and starts at 0 (zero).

This attribute is writeable, in order to allow setting to an appropriate value after maintenance.

Valid range is 0 kWh to 4,294,967,294 kWh. The value 4,294,967,295 (0xffffffff) indicates that this value is unknown.

6.2.2.2.3 Pump Settings Attribute Set

The pump settings attribute set contains the attributes summarized in Table 6-6:

Table 6-6. Attributes of the Pump Settings Attribute Set

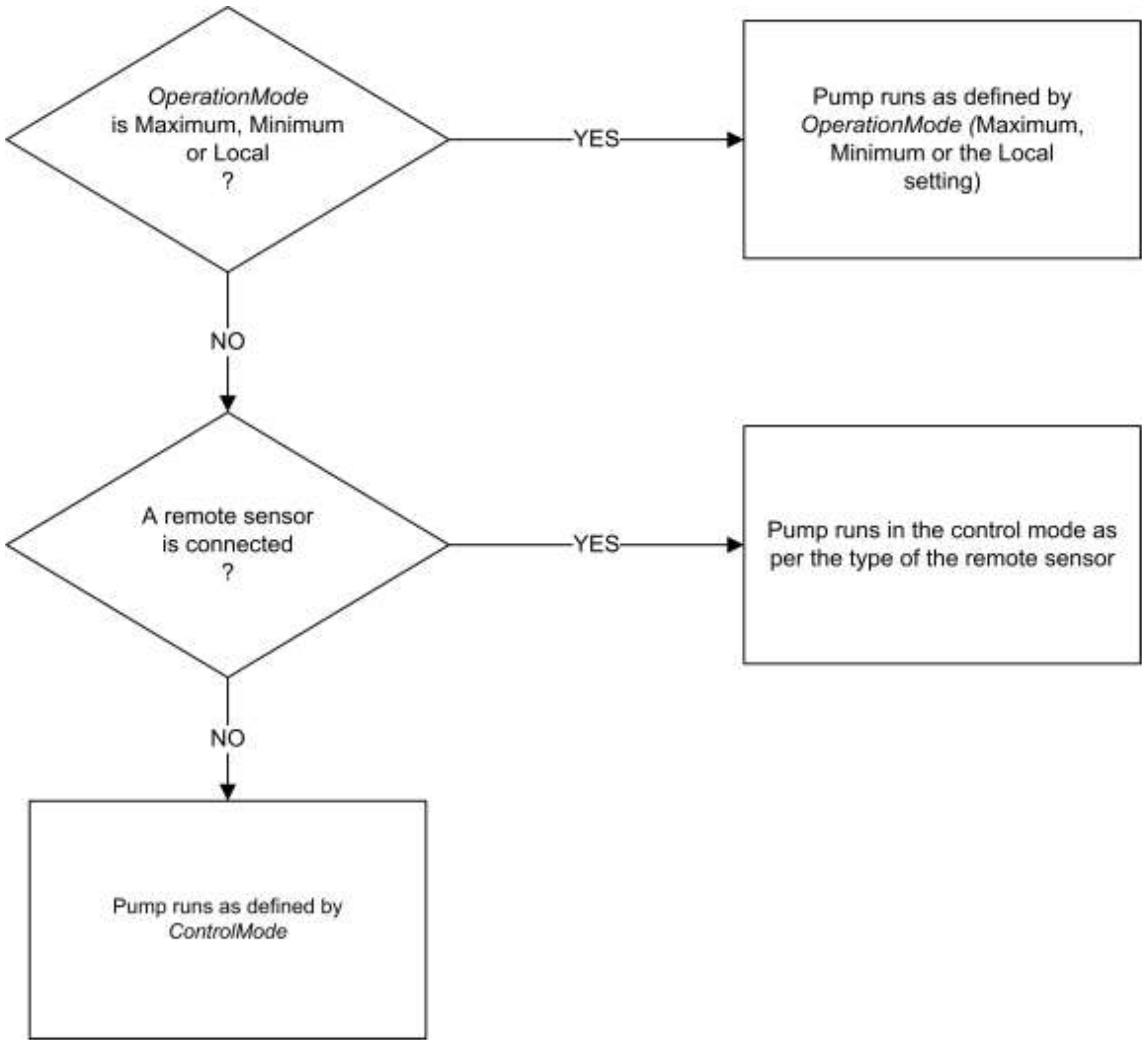
Identifier	Name	Type	Range	Access	Def	M/O
0x0020	*OperationMode*	enum8	0x00 – 0xfe	RW	0x00	M
0x0021	*ControlMode*	enum8	0x00 – 0xfe	RW	0x00	O
0x0022	*AlarmMask*	map16	-	R	-	O

6.2.2.2.3.1 *OperationMode* Attribute

The *OperationMode* attribute specifies the operation mode of the pump. This attribute SHALL have one of the values listed in Table 6-7Values of the .

The actual operating mode of the pump is a result of the setting of the attributes *OperationMode*, *ControlMode* and the optional connection of a remote sensor. The operation and control is prioritized as shown in the scheme in Figure 6-3:

Figure 6-3. Priority Scheme of Pump Operation and Control


OperationMode is Maximum, Minimum or Local ?
YES
Pump runs as defined by OperationMode (Maximum, Minimum or the Local setting)
NO
A remote sensor is connected ?
YES
Pump runs in the control mode as per the type of the remote sensor
NO
Pump runs as defined by ControlMode

If the *OperationMode* attribute is Maximum, Minimum or Local, the *OperationMode* attribute decides how the pump is operated.

If the *OperationMode* attribute is Normal and a remote sensor is connected to the pump, the type of the remote sensor decides the control mode of the pump. A connected remote pressure sensor will make the pump run in control mode Constant pressure and vice versa for flow and temperature type sensors. This is regardless of the setting of the *ControlMode* attribute.

If the *OperationMode* attribute is Normal and no remote sensor is connected, the control mode of the pump is decided by the *ControlMode* attribute.

OperationMode MAY be changed at any time, even when the pump is running. The behavior of the pump at the point of changing the value of the *OperationMode* attribute is vendor-specific.

Table 6-7. Values of the *OperationMode* Attribute

Value	Name	Explanation
0	Normal	The pump is controlled by a setpoint, as defined by a connected remote sensor or by the *ControlMode* attribute. (N.B. The setpoint is an internal variable which MAY be controlled between 0% and 100%, e.g., by means of the Level Control cluster 3.10)
1	Minimum	This value sets the pump to run at the minimum possible speed it can without being stopped
2	Maximum	This value sets the pump to run at its maximum possible speed
3	Local	This value sets the pump to run with the local settings of the pump, regardless of what these are

6.2.2.2.3.2 *ControlMode* Attribute

The *ControlMode* attribute specifies the control mode of the pump. This attribute SHALL have one of the values listed in Table 6-8Values of the .

See section 6.2.2.2.3.1 for detailed description of the operation and control of the pump.

ControlMode MAY be changed at any time, even when the pump is running. The behavior of the pump at the point of changing is vendor-specific.

Table 6-8. Values of the *ControlMode* Attribute

Value	Name	Explanation
0	Constant speed	The pump is running at a constant speed. The setpoint is interpreted as a percentage of the *MaxSpeed* attribute
1	Constant pressure	The pump will regulate its speed to maintain a constant differential pressure over its flanges. The setpoint is interpreted as a percentage of the range of the sensor used for this control mode. In case of the internal pressure sensor, this will be the range derived from the [*MinConstPressure* - *MaxConstPressure*] attributes. In case of a remote pressure sensor, this will be the range derived from the [*MinMeasuredValue* – *MaxMeasuredValue*] attributes of the remote pressure sensor.
2	Proportional pressure	The pump will regulate its speed to maintain a constant differential pressure over its flanges. The setpoint is interpreted as a percentage of the range derived of the [*MinCompPressure* - *MaxCompPressure*] attributes. The internal setpoint will be lowered (compensated) dependant on the flow in the pump (lower flow => lower internal setpoint)
3	Constant flow	The pump will regulate its speed to maintain a constant flow through the pump. The setpoint is interpreted as a percentage of the range of the sensor used for this control mode. In case of the internal flow sensor, this will be the range derived from the [*MinConstFlow* - *MaxConstFlow*] attributes. In case of a remote flow sensor, this will be the range derived from the [*MinMeasuredValue* – *MaxMeasuredValue*] attributes of the remote flow sensor.
5	Constant temperature	The pump will regulate its speed to maintain a constant temperature. The setpoint is interpreted as a percentage of the range of the sensor used for this control mode. In case of the internal temperature sensor, this will be the range derived from the [*MinConstTemp* - *MaxConstTemp*] attributes. In case of a remote temperature sensor, this will be the range derived from the [*MinMeasuredValue* – *MaxMeasuredValue*] attributes of the remote temperature sensor.

Value	Name	Explanation
7	Automatic	The operation of the pump is automatically optimized to provide the most suitable performance with respect to comfort and energy savings. This behavior is manufacturer defined. The pump can be stopped by setting the setpoint of the level control cluster to 0 of by using the On/Off cluster. If the pump is started (at any setpoint), the speed of the pump is entirely determined by the pump.

6.2.2.2.3.3 *AlarmMask* Attribute

The *AlarmMask* attribute specifies whether each of the alarms listed in Table 6-9 is enabled. When the bit number corresponding to the alarm code is set to 1, the alarm is enabled, else it is disabled. Bits not corresponding to a code in the table (bits 14, 15) are reserved.

When the Alarms cluster is implemented on a device, and one of the alarm conditions included in this table occurs, an alarm notification is generated, with the alarm code field set as listed in the table.

Table 6-9. Alarm Codes

Alarm Code	Alarm Condition
0	Supply voltage too low
1	Supply voltage too high
2	Power missing phase
3	System pressure too low
4	System pressure too high
5	Dry running
6	Motor temperature too high
7	Pump motor has fatal failure
8	Electronic temperature too high
9	Pump blocked
10	Sensor failure
11	Electronic non fatal failure
12	Electronic fatal failure
13	General fault

6.2.2.3 Commands

The server does not receive or generate cluster specific commands.

6.2.2.4 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes command, according to the minimum and maximum reporting interval, reportable change, and timeout period settings described in the ZCL Foundation Specification (see 2.4.7).

The following attributes SHALL be reported:

PumpStatus
Capacity

6.2.3 Client

The client supports no specific attributes. The client does not receive or generate cluster specific commands.

6.3 Thermostat

6.3.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface to the functionality of a thermostat.

6.3.1.1 Revision History

Rev	Description
1	mandatory global ClusterRevision attribute added; fixed some defaults; CCB 1823, 1480
2	CCB 1981 2186 2249 2250 2251; NFR Thermostat Setback

6.3.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	TSTAT	Type 2 (server to client)

6.3.1.3 Cluster Identifiers

Identifier	Name
0x0201	Thermostat

6.3.1.4 Thermostat Temperature Conversion

Many Thermostats store internally or have the capability to display the temperature in degree Fahrenheit format. The Thermostat cluster standardizes temperature representation in degree Celsius format when transferred over the air. Sample code has been provided (see 6.6.2.3). Manufacturers SHOULD use the conversion algorithm provided to convert temperature from Fahrenheit to Celsius and vice versa.

6.3.1.5 Thermostat Schedule Feature Mandatory Requirement

The *StartOfWeek* Attribute is the indicator to show that the Weekly schedule extension is supported. If the Weekly schedule extension feature is supported, it is mandatory to also support the *StartOfWeek* Attribute, *NumberOfWeeklyTransitions* Attribute, *NumberOfDailyTransitions* Attribute, Set Weekly Schedule Command and Get Weekly Schedule Command.

6.3.2 Server

6.3.2.1 Dependencies

For alarms to be generated by this cluster, the Alarms server cluster (see 3.11) SHALL be included on the same endpoint. For remote temperature sensing, the Temperature Measurement client cluster (see 4.4) MAY be included on the same endpoint. For occupancy sensing, the Occupancy Sensing client cluster (see 4.8) MAY be included on the same endpoint.

6.3.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets for Thermostat are listed in Table 6-10.

Table 6-10. Currently Defined Thermostat Attribute Sets

Attribute Set Identifier	Description
0x000	Thermostat Information
0x001	Thermostat Settings
0x002[104]	Thermostat Schedule & HVAC Relay Attribute Set
0x003	Thermostat Setpoint Change Tracking Attribute Set
0x004	AC Information Attribute Set
0x400 – 0xfff	Reserved for vendor specific attributes

6.3.2.2.1 Thermostat Information Attribute Set

The Thermostat Information attribute set contains the attributes summarized in Table 6-11.

Table 6-11. Attributes of the Thermostat Information Attribute Set

Id	Name	Type	Range	Acc	Default	M/O
0x0000	*LocalTemperature*	int16	0x954d – 0x7fff	RP	-	M
0x0001	*OutdoorTemperature*	int16	0x954d – 0x7fff	R	-	O
0x0002	*Occupancy*	map8	0000000x	R	00000000	O

[104] CCB 2251 added missing sets 0x002-0x004

Id	Name	Type	Range	Acc	Default	M/O
0x0003	*AbsMinHeatSetpointLimit*	int16	0x954d – 0x7fff	R	0x02bc (7°C)	O
0x0004	*AbsMaxHeatSetpointLimit*	int16	0x954d – 0x7fff	R	0x0bb8 (30°C)	O
0x0005	*AbsMinCoolSetpointLimit*	int16	0x954d – 0x7fff	R	0x0640 (16°C)	O
0x0006	*AbsMaxCoolSetpointLimit*	int16	0x954d – 0x7fff	R	0x0c80 (32°C)	O
0x0007	*PICoolingDemand*	uint8	0x00 – 0x64	RP	-	O
0x0008	*PIHeatingDemand*	uint8	0x00 – 0x64	RP	-	O
0x0009	*HVACSystemTypeConfiguration*	map8	00xxxxxx	RW	00000000	O

6.3.2.2.1.1 *LocalTemperature* Attribute

LocalTemperature represents the temperature in degrees Celsius, as measured locally or remotely (over the network), including any adjustments applied by *LocalTemperatureCalibration* attribute (if any)[105] as follows:

LocalTemperature = 100 x (temperature in degrees Celsius + *LocalTemperatureCalibration).*

Where -273.15°C <= temperature <= 327.67 ºC, corresponding to a *LocalTemperature* in the range 0x954d to 0x7fff.

The maximum resolution this format allows is 0.01 ºC.

A *LocalTemperature* of 0x8000 indicates that the temperature measurement is invalid.

6.3.2.2.1.2 All setpoint attributes in the Thermostat cluster SHALL be triggered based off the *LocalTemperature* attribute (i.e., measured temperature and any calibration offset). *OutdoorTemperature* Attribute

OutdoorTemperature represents the outdoor temperature in degrees Celsius, as measured locally or remotely (over the network). It is measured as described for *LocalTemperature.*

6.3.2.2.1.3 *Occupancy* Attribute

Occupancy specifies whether the heated/cooled space is occupied or not, as measured locally or remotely (over the network). If bit 0 = 1, the space is occupied, else it is unoccupied. All other bits are reserved.

6.3.2.2.1.4 *AbsMinHeatSetpointLimit* Attribute

The *MinHeatSetpointLimit* attribute specifies the absolute minimum level that the heating setpoint MAY be set to. This is a limitation imposed by the manufacturer. The value is calculated as described in the LocalTemperature attribute.

6.3.2.2.1.5 *AbsMaxHeatSetpointLimit* Attribute

The *MaxHeatSetpointLimit* attribute specifies the absolute maximum level that the heating setpoint MAY be set to. This is a limitation imposed by the manufacturer. The value is calculated as described in the LocalTemperature attribute.

6.3.2.2.1.6 *AbsMinCoolSetpointLimit* Attribute

The *MinCoolSetpointLimit* attribute specifies the absolute minimum level that the cooling setpoint MAY be set to. This is a limitation imposed by the manufacturer. The value is calculated as described in the LocalTemperature attribute.

[105] NFR Thermostat Setback

6.3.2.2.1.7 *AbsMaxCoolSetpointLimit* Attribute

The *MaxCoolSetpointLimit* attribute specifies the absolute maximum level that the cooling setpoint MAY be set to. This is a limitation imposed by the manufacturer. The value is calculated as described in the LocalTemperature attribute.

6.3.2.2.1.8 *PICoolingDemand* Attribute

The *PICoolingDemand* attribute is 8 bits in length and specifies the level of cooling demanded by the PI (proportional integral) control loop in use by the thermostat (if any), in percent. This value is 0 when the thermostat is in “off” or “heating” mode.

This attribute is reported regularly and MAY be used to control a heating device.

6.3.2.2.1.9 *PIHeatingDemand* Attribute

The *PIHeatingDemand* attribute is 8 bits in length and specifies the level of heating demanded by the PI loop in percent. This value is 0 when the thermostat is in “off” or “cooling” mode.

This attribute is reported regularly and MAY be used to control a cooling device.

6.3.2.2.1.10 HVACSystemTypeConfiguration Attribute

The *HVACSystemTypeConfiguration* attribute specifies the HVAC system type controlled by the thermostat. If the thermostat uses physical DIP switches to set these parameters, this information SHALL be available read-only from the DIP switches. If these parameters are set via software, there SHALL be read/write access in order to provide remote programming capability. The meanings of individual bits are detailed in Table 6-12. Each bit represents a type of system configuration.

Table 6-12. HVAC System Type Configuration Values

Bit Number	Description
0 – 1	**Cooling System Stage** 00 – Cool Stage 1 01 – Cool Stage 2 10 – Cool Stage 3 11 – Reserved
2 – 3	**Heating System Stage** 00 – Heat Stage 1 01 – Heat Stage 2 10 – Heat Stage 3 11 – Reserved
4	**Heating System Type** 0 – Conventional 1 – Heat Pump
5	**Heating Fuel Source** 0 – Electric / B 1 – Gas / O

6.3.2.2.2 Thermostat Settings Attribute Set

The Thermostat settings attribute set contains the attributes summarized in Table 6-13:

Table 6-13. Attributes of the Thermostat Settings Attribute Set

Id	Name	Type	Range	Acc	Def	MO
0x0010	*LocalTemperatureCalibration*	int8	0xE7 – 0x19	RW	0x00 (0°C)	O
0x0011	*OccupiedCoolingSetpoint*	int16	*MinCoolSetpointLimit* to *MaxCoolSetpointLimit*	RWS	0x0a28 (26°C)	M*
0x0012	*OccupiedHeatingSetpoint*	int16	*MinHeatSetpointLimit* to *MaxHeatSetpointLimit*	RWS	0x07d0 (20°C)	M*
0x0013	*UnoccupiedCoolingSetpoint*	int16	*MinCoolSetpointLimit* to *MaxCoolSetpointLimit*	RW	0x0a28 (26°C)	O
0x0014	*UnoccupiedHeatingSetpoint*	int16	*MinHeatSetpointLimit* to *MaxHeatSetpointLimit*	RW	0x07d0 (20°C)	O
0x0015	*MinHeatSetpointLimit*	int16	0x954d – 0x7fff	RW	0x02bc (7°C)	O
0x0016	*MaxHeatSetpointLimit*	int16	0x954d – 0x7fff	RW	0x0bb8 (30°C)	O
0x0017	*MinCoolSetpointLimit*	int16	0x954d – 0x7fff	RW	0x0640 (16°C)	O
0x0018	*MaxCoolSetpointLimit*	int16	0x954d – 0x7fff	RW	0x0c80 (32°C)	O
0x0019	*MinSetpointDeadBand*	int8	0x0a – 0x19	RW	0x19 (2.5°C)	O
0x001a	*RemoteSensing*	map8	00000xxx	RW	0	O
0x001b	*ControlSequenceOfOperation*	enum8	0x00 – 0x05	RW	0x04	M
0x001c	*SystemMode*	enum8	See Table 6-16	RWS	0x01	M
0x001d	*AlarmMask*	map8	00000xxx	R	0	O
0x001e	*ThermostatRunningMode*	enum8	0x00 – 0x04	R	0x00	O

**Note:* "M*" designates that a server SHALL implement at least one of the attributes designated "M*." For example, a radiator valve implementing the Thermostat Cluster server would only implement the OccupiedHeatingSetpoint attribute. Thermostats SHOULD implement both OccupiedCoolingSetpoint and OccupiedHeatingSetpoint attributes. The "M*" designation allows HVAC devices to implement the portions of Thermostat cluster germane to their operation.*

6.3.2.2.2.1 *LocalTemperatureCalibration* Attribute

Specifies the offset the Thermostat server SHALL make to the measured temperature (locally or remotely) before calculating, displaying, or communicating the *LocalTemperature* attribute, in steps of 0.1°C.

The purpose of this attribute is to adjust the calibration of the Thermostat server per the user's preferences (e.g., to match if there are multiple servers displaying different values for the same HVAC area) or compensate for variability amongst temperature sensors.

If a Thermostat client attempts to write *LocalTemperatureCalibration* attribute to an unsupported value (e.g., out of the range supported by the Thermostat server), the Thermostat server SHALL respond with a Write Attribute Response Command with a status of LIMIT_REACHED and set the value of *LocalTemperatureCalibration* to the upper or lower limit reached.[106]

6.3.2.2.2.2 *OccupiedCoolingSetpoint* Attribute

The *OccupiedCoolingSetpoint* attribute specifies the cooling mode setpoint when the room is occupied

The *OccupiedHeatingSetpoint* attribute SHALL always be below the value specified in the *OccupiedCooling Setpoint* by at least *MinSetpointDeadband*[107]. If an attempt is made to set it such that this condition is violated, a response command with the status code INVALID_VALUE (see 2.5.3) SHALL be returned. If the occupancy status of the room is unknown, this attribute SHALL be used as the cooling mode setpoint.

6.3.2.2.2.3 *OccupiedHeatingSetpoint* Attribute

The *OccupiedHeatingSetpoint* attribute specifies the heating mode setpoint when the room is occupied. The *OccupiedCoolingSetpoint* attribute SHALL always be above the value specified in the *OccupiedHeatingSetpoint* by at least *MinSetpointDeadband*.

If the occupancy status of the room is unknown, this attribute SHALL be used as the heating[108] mode setpoint.

6.3.2.2.2.4 *UnoccupiedCoolingSetpoint* Attribute

The *UnoccupiedCoolingSetpoint* attribute and specifies the cooling mode setpoint when the room is unoccupied. The *UnoccupiedHeatingSetpoint* attribute SHALL always be below the value specified in the *UnoccupiedCoolingSetpoint* by at least *MinSetpointDeadband*.

If the occupancy status of the room is unknown, this attribute SHALL not be used.

6.3.2.2.2.5 *UnoccupiedHeatingSetpoint* Attribute

The *UnoccupiedHeatingSetpoint* attribute specifies the heating mode setpoint when the room is unoccupiedThe *UnoccupiedCoolingSetpoint* attribute SHALL always be above the value specified in the *UnoccupiedHeatingSetpoint* by at least *MinSetpointDeadband*.

If the occupancy status of the room is unknown, this attribute SHALL not be used.

6.3.2.2.2.6 *MinHeatSetpointLimit* Attribute

The *MinHeatSetpointLimit* attribute specifies the minimum level that the heating setpoint MAY be set to. If this attribute is not present, it SHALL be taken as equal to *AbsMinHeatSetpointLimit*.

This attribute, and the following three attributes, allow the user to define setpoint limits more constrictive than the manufacturer imposed ones. Limiting users (e.g., in a commercial building) to such setpoint limits can help conserve power.

6.3.2.2.2.7 *MaxHeatSetpointLimit* Attribute

The *MaxHeatSetpointLimit* attribute specifies the maximum level that the heating setpoint MAY be set to. It must be less than or equal to *AbsMaxHeatSetpointLimit*. If this attribute is not present, it SHALL be taken as equal to *AbsMaxHeatSetpointLimit*.

6.3.2.2.2.8 *MinCoolSetpointLimit* Attribute

[106] CCB 1981 & NFR Thermostat Setback
[107] CCB 2251 nisnamed *SetpointDeadBand (and other places as well)*
[108] CCB 2186 typo: should be used for heating (not cooling)

The *MinCoolSetpointLimit* attribute specifies the minimum level that the cooling setpoint MAY be set to. It must be greater than or equal to *AbsMinCoolSetpointLimit*. If this attribute is not present, it SHALL be taken as equal to *AbsMinCoolSetpointLimit*.

6.3.2.2.2.9 *MaxCoolSetpointLimit* Attribute

The *MaxCoolSetpointLimit* attribute specifies the maximum level that the cooling setpoint MAY be set to. . It must be less than or equal to *AbsMaxCoolSetpointLimit*. If this attribute is not present, it SHALL be taken as equal to *AbsMaxCoolSetpointLimit*.

6.3.2.2.2.10 *MinSetpointDeadBand* Attribute

The *MinSetpointDeadBand* attribute specifies the minimum difference between the Heat Setpoint and the Cool SetPoint, in steps of 0.1°C. Its range is 0x0a to 0x19 (1°C to 2.5°C).

6.3.2.2.2.11 *RemoteSensing* Attribute

The RemoteSensing attribute is an 8-bit bitmap that specifies whether the local temperature, outdoor temperature and occupancy are being sensed by internal sensors or remote networked sensors. The meanings of individual bits are detailed in Table 6-14.

Table 6-14. *RemoteSensing* Attribute Bit Values

Bit Number	Description
0	0 – local temperature sensed internally 1 – local temperature sensed remotely
1	0 – outdoor temperature sensed internally 1 – outdoor temperature sensed remotely
2	0 – occupancy sensed internally 1 – occupancy sensed remotely

6.3.2.2.2.12 *ControlSequenceOfOperation* Attribute

The *ControlSequenceOfOperation* attribute specifies the overall operating environment of the thermostat, and thus the possible system modes that the thermostat can operate in. It SHALL be set to one of the non-reserved values in Table 6-15. (**Note:** it is not mandatory to support all values).

Table 6-15. *ControlSequenceOfOperation* Attribute Values

Value	Description	Possible Values of *SystemMode*
0x00	Cooling Only	Heat and Emergency are not possible
0x01	Cooling With Reheat	Heat and Emergency are not possible
0x02	Heating Only	Cool and precooling (see 6.1.2) are not possible
0x03	Heating With Reheat	Cool and precooling are not possible
0x04	Cooling and Heating 4-pipes (see 1.3.2)	All modes are possible
0x05	Cooling and Heating 4-pipes with Reheat	All modes are possible

6.3.2.2.2.13 *SystemMode* Attribute

The *SystemMode* attribute specifies the current operating mode of the thermostat, It SHALL be set to one of the non-reserved values in Table 6-16, as limited by Table 6-17. (**Note:** It is not mandatory to support all values).

Table 6-16. *SystemMode* Attribute Values

Attribute Value	Description
0x00	Off
0x01	Auto
0x03	Cool
0x04	Heat
0x05	Emergency heating
0x06	Precooling (see 6.1.2)
0x07	Fan only
0x08	Dry
0x09	Sleep

The interpretation of the Heat, Cool and Auto values of *SystemMode* is shown in Table 6-17.

Table 6-17. Interpretation of *SystemMode* Values

Attribute Values	Temperature Below Heat Setpoint	Temperature Between Heat Setpoint and Cool Setpoint	Temperature Above Cool Setpoint
Heat	Temperature below target	Temperature on target	Temperature on target
Cool	Temperature on target	Temperature on target	Temperature above target
Auto	Temperature below target	Temperature on target	Temperature above target

6.3.2.2.2.14 *AlarmMask* Attribute

The *AlarmMask* attribute specifies whether each of the alarms listed in Table 6-18Alarm Codes is enabled. When the bit number corresponding to the alarm code is set to 1, the alarm is enabled, else it is disabled. Bits not corresponding to a code in the table are reserved.

When the Alarms cluster is implemented on a device, and one of the alarm conditions included in this table occurs, an alarm notification is generated, with the alarm code field set as listed in the table.

Table 6-18. Alarm Codes

Alarm Code	Alarm Condition
0	Initialization failure. The device failed to complete initialization at power-up.
1	Hardware failure
2	Self-calibration failure

6.3.2.2.2.15 Thermostat Running Mode Attribute

ThermostatRunningMode represents the running mode of the thermostat. The thermostat running mode can only be Off, Cool or Heat. This attribute is intended to provide additional information when the thermostat's system mode is in auto mode. The attribute value is maintained to have the same value as the *SystemMode* attribute.

Table 6.19 Thermostat Running Mode Attribute Values

Value	Description
0x00	Off
0x03	Cool
0x04	Heat

6.3.2.2.3 Thermostat Schedule & HVAC Relay Attribute Set

Table 6-20. Thermostat Schedule & HVAC Relay Attribute Set

Id	Name	Type	Range	Access	Default	M/O
Schedule Attribute Set 0x0020 – 0x0028						
x0020	*StartOfWeek*	enum8	0x00 – 0x06	R	–	O
x0021	*NumberOfWeeklyTransitions*	uint8	0x00 – 0xff	R	N/A	O
x0022	*NumberOfDailyTransitions*	uint8	0x00 – 0xff	R	N/A	O
x0023	*TemperatureSetpointHold*	enum8	0x00 – 0x01	RW	0x00	O
x0024	*TemperatureSetpointHoldDuration*	uint16	0xffff - 0x05a0	RW	0xffff	O
x0025	*ThermostatProgrammingOperationMode*	map8	00xxxxxx	RWP[109]	00000000	O
HVAC Relay Attribute Set 0x0029 – 0x002F						
x0029	*ThermostatRunningState*	map16		R	-	O

6.3.2.2.3.1 StartOfWeek Attribute

StartOfWeek represents the day of the week that this thermostat considers to be the start of week for weekly set point scheduling. The possible values are given in Table 6-21:

Table 6-21. *StartofWeek* Enumeration Values

Enumeration Field	Value Description
0x00	Sunday
0x01	Monday
0x02	Tuesday
0x03	Wednesday
0x04	Thursday

[109] CCB 2250 attribute specification text requires that this be reportable

Enumeration Field	Value Description
0x05	Friday
0x06	Saturday

If the Weekly schedule extension is supported this attribute SHALL be supported.

This attribute MAY be able to be used as the base to determine if the device supports weekly scheduling by reading the attribute. Successful response means that the weekly scheduling is supported.

6.3.2.2.3.2 NumberOfWeeklyTransitions Attribute

NumberOfWeeklyTransitions attribute determines how many weekly schedule transitions the thermostat is capable of handling.

6.3.2.2.3.3 NumberOfDailyTransitions Attribute

NumberOfDailyTransitions attribute determines how many daily schedule transitions the thermostat is capable of handling.

6.3.2.2.3.4 TemperatureSetpointHold Attribute

TemperatureSetpointHold specifies the temperature hold status on the thermostat, as shown in Table 6-22. If hold status is on, the thermostat SHOULD maintain the temperature set point for the current mode until a system mode change. If hold status is off, the thermostat SHOULD follow the setpoint transitions specified by its internal scheduling program. If the thermostat supports setpoint hold for a specific duration, it SHOULD also implement the *TemperatureSetpointHoldDuration* attribute.

Table 6-22. *TemperatureSetpointHold* Attribute Values

Enumeration Field	Value Description
0x00	Setpoint Hold Off
0x01	Setpoint Hold On

6.3.2.2.3.5 TemperatureSetpointHoldDuration Attribute

TemperatureSetpointHoldDuration sets the period in minutes for which a setpoint hold is active. Thermostats that support hold for a specified duration SHOULD implement this attribute. The valid range is from 0x0000 – 0x05A0 (1440 minutes within a day). A value of 0xFFFF indicates the field is unused. All other values are reserved.

6.3.2.2.3.6 ThermostatProgrammingOperationMode Attribute

The *ThermostatProgrammingOperation*Mode attribute determines the operational state of the thermostat's programming. The thermostat SHALL modify its programming operation when this attribute is modified by a client and update this attribute when its programming operation is modified locally by a user. The thermostat MAY support more than one active *ThermostatProgrammingOperationMode*. For example, the thermostat MAY operate simultaneously in Schedule Programming Mode and Recovery Mode. If a thermostat supports Thermostat Programming Operation Mode attribute, it SHALL support attribute reporting for this attribute. Any locally-initiated changes to the *ThermostatProgrammingOperationMode* SHALL be updated and reported to all clients configured to receive such reports. The meanings of individual bits are detailed in Table 6-23. Each bit represents a type of operation.

Table 6-23. *ThermostatProgrammingOperationMode* Attribute Values

Value	Description
0	0 – Simple/setpoint mode. This mode means the thermostat setpoint is altered only by manual up/down changes at the thermostat or remotely, not by internal schedule programming. 1 – Schedule programming mode. This enables or disables any programmed weekly schedule configurations. ***Note:*** *It does not clear or delete previous weekly schedule programming configurations.*
1	0 - Auto/recovery mode set to OFF 1 – Auto/recovery mode set to ON
2	0 – Economy/EnergyStar mode set to OFF 1 – Economy/EnergyStar mode set to ON

6.3.2.2.3.7 ThermostatRunningState Attribute

ThermostatRunningState represents the current relay state of the heat, cool, and fan relays, whose values are shown in Table 6-24.

Table 6-24. HVAC Relay State Values

Bit Number	Description
0	Heat State On
1	Cool State On
2	Fan State On
3	Heat 2nd Stage State On
4	Cool 2nd Stage State On
5	Fan 2nd Stage State On
6	Fan 3rd Stage Stage On

6.3.2.2.4 Thermostat Setpoint ChangeTracking Attribute Set

Table 6-25. Thermostat Setpoint Change Tracking Attribute Set

Id	Name	Type	Range	Acc	Def	MO
0x0030	*SetpointChangeSource*	enum8	0x00 – 0xff	R	0x00	O
0x0031	*SetpointChangeAmount*	int16	0x0000 – 0xffff	R	0x8000	O
0x0032	*SetpointChangeSourceTimestamp*	UTC[110]	0x00000000 – 0xfffffffe	R	0x00000000	O
0x0034	*OccupiedSetback*[111]	uint8	*OccupiedSetbackMin – OccupiedSetbackMax*	RW	0xff	O

[110] CCB 2249 was listed in table as int32 (UTC), though specified as UTC in text
[111111] NFR Thermostat Setback attributes 0x0035-0x003a

Id	Name	Type	Range	Acc	Def	MO
0x0035	*OccupiedSetbackMin*	uint8	0x00 – *OccupiedSetbackMax*	R	0xff	O
0x0036	*OccupiedSetbackMax*	uint8	*OccupiedSetbackMin* – 0xff	R	0xff	O
0x0037	*UnoccupiedSetback*	uint8	*UnoccupiedSetbackMin* – *OccupiedSetbackMax*	RW	0xff	O
0x0038	*UnoccupiedSetbackMin*	uint8	0x00 – *OccupiedSetbackMax*	R	0xff	O
0x0039	*UnoccupiedSetbackMax*	uint8	*OccupiedSetbackMin* – 0xff	R	0xff	O
0x003a	*EmergencyHeatDelta*	uint8	0x00 – 0xff	RW	0xff	O

6.3.2.2.4.1 SetpointChangeSource Attribute

The *SetpointChangeSource* attribute specifies the source of the current active *OccupiedCoolingSetpoint* or *OccupiedHeatingSetpoint* (i.e., who or what determined the current setpoint).

SetpointChangeSource attribute enables service providers to determine whether changes to setpoints were initiated due to occupant comfort, scheduled programming or some other source (e.g., electric utility or other service provider). Because automation services MAY initiate frequent setpoint changes, this attribute clearly differentiates the source of setpoint changes made at the thermostat.

Table 6-26. *SetpointChangeSource* Values

***SetpointChangeSource* Attribute**	**Description**
0x00	Manual, user-initiated setpoint change via the thermostat
0x01	Schedule/internal programming-initiated setpoint change
0x02	Externally-initiated setpoint change (e.g., DRLC cluster command, attribute write)

6.3.2.2.4.2 SetpointChangeAmount Attribute

The *SetpointChangeAmount* attribute specifies the delta between the current active *OccupiedCoolingSetpoint* or *OccupiedHeatingSetpoint* and the previous active setpoint. This attribute is meant to accompany the *SetpointChangeSource* attribute; devices implementing *SetpointChangeAmount* SHOULD also implement *SetpointChangeSource*.

Table 6-27. *SetpointChangeAmount* Values

***SetpointChangeAmount* Attribute**	**Description**
0x0000 – 0xffff	The signed difference in 0.01 degrees Celsius between the previous temperature setpoint and the new temperature setpoint.

6.3.2.2.4.3 SetpointChangeSourceTimestamp Attribute

The *SetpointChangeSourceTimestamp* attribute specifies the time in UTC at which the *SetpointChangeSourceAmount* attribute change was recorded.

6.3.2.2.4.4 OccupiedSetback Attribute

Specifies the degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the *LocalTemperature* attribute to float above the *OccupiedCooling* setpoint (i.e., *OccupiedCooling* + *OccupiedSetback*) or below the *OccupiedHeating* setpoint (i.e., *OccupiedHeating* – *OccupiedSetback*) before initiating a state change to bring the temperature back to the user's desired setpoint. This attribute is sometimes also referred to as the "span."

The purpose of this attribute is to allow remote configuration of the span between the desired setpoint and the measured temperature to help prevent over-cycling and reduce energy bills, though this may result in lower comfort on the part of some users.

A value of 0xff indicates the attribute is unused.

If *OccupiedSetback* is implemented, then the Thermostat server SHALL also implement *OccupiedSetbackMin* and *OccupiedSetbackMax* attributes.

If the Thermostat client attempts to write *OccupiedSetback* to a value greater than *OccupiedSetbackMax*, the Thermostat server SHALL set its *OccupiedSetback* value to *OccupiedSetbackMax* and SHALL send a Write Attribute Response command with a Status Code field enumeration of LIMIT_REACHED response.

If the Thermostat client attempts to write *OccupiedSetback* to a value less than *OccupiedSetbackMin*, the Thermostat server SHALL set its *OccupiedSetback* value to *OccupiedSetbackMin* and SHALL send a Write Attribute Response command with a Status Code field enumeration of LIMIT_REACHED response.

6.3.2.2.4.5 OccupiedSetbackMin Attribute

Specifies the minimum degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the *OccupiedSetback* attribute to be configured by a user.

A value of 0xff indicates the attribute is unused.

OccupiedSetbackMin attribute value SHALL be less than *OccupiedSetbackMax* attribute value. Attempts to configure *OccupiedSetbackMin* with a value greater than or equal to the value of the *OccupiedSetbackMax* attribute SHALL cause the Thermostat server to respond with a Write Attribute Response Command containing the status INVALID_VALUE and SHALL revert back to the previous *OccupiedSetbackMin* attribute value.

If *OccupiedSetbackMin* is configured to a value greater than the value of *OccupiedSetback*, then the Thermostat server SHALL update the value of *OccupiedSetback* to equal the new value of *OccupiedSetbackMin*.

6.3.2.2.4.6 OccupiedSetbackMax Attribute

Specifies the maximum degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the *OccupiedSetback* attribute to be configured by a user.

A value of 0xff indicates the attribute is unused.

OccupiedSetbackMax attribute value SHALL be greater than *OccupiedSetbackMin* attribute value. Attempts to configure *OccupiedSetbackMax* with a value less than or equal to the value of the *OccupiedSetbackMin* attribute SHALL cause the Thermostat server to respond with a Write Attribute Response Command containing the status INVALID_VALUE and SHALL revert back to the previous *OccupiedSetbackMax* attribute value.

If *OccupiedSetbackMax* is configured to a value less than the value of *OccupiedSetback*, then the Thermostat server SHALL update the value of *OccupiedSetback* to equal the new value of *OccupiedSetbackMax*.

6.3.2.2.4.7 UnoccupiedSetback Attribute

Specifies the degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the LocalTemperature attribute to float above the UnoccupiedCooling setpoint (i.e., UnoccupiedCooling + UnoccupiedSetback) or below the UnoccupiedHeating setpoint (i.e., UnoccupiedHeating - UnoccupiedSetback) before initiating a state change to bring the temperature back to the user's desired setpoint. This attribute is sometimes also referred to as the "span."

The purpose of this attribute is to allow remote configuration of the span between the desired setpoint and the measured temperature to help prevent over-cycling and reduce energy bills, though this may result in lower comfort on the part of some users.

A value of 0xff indicates the attribute is unused.

If UnoccupiedSetback is implemented, then the Thermostat server SHALL also implement UnoccupiedSetbackMin and UnoccupiedSetbackMax attributes.

If the Thermostat client attempts to write UnoccupiedSetback to a value greater than UnoccupiedSetbackMax, the Thermostat server SHALL set its UnoccupiedSetback value to UnoccupiedSetbackMax and SHALL send a Write Attribute Response command with a Status Code field enumeration of LIMIT_REACHED response.

If the Thermostat client attempts to write UnoccupiedSetback to a value less than UnoccupiedSetbackMin, the Thermostat server SHALL set its UnoccupiedSetback value to UnoccupiedSetbackMin and SHALL send a Write Attribute Response command with a Status Code field enumeration of LIMIT_REACHED response.

6.3.2.2.4.8 UnoccupiedSetbackMin Attribute

Specifies the minimum degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the *UnoccupiedSetback* attribute to be configured by a user.

A value of 0xff indicates the attribute is unused.

UnoccupiedSetbackMin attribute value SHALL be less than *UnoccupiedSetbackMax* attribute value. Attempts to configure *UnoccupiedSetbackMin* with a value greater than or equal to the value of the *UnoccupiedSetbackMax* attribute SHALL cause the Thermostat server to respond with a Write Attribute Response Command containing the status INVALID_VALUE and SHALL revert back to the previous *UnoccupiedSetbackMin* attribute value.

If *UnoccupiedSetbackMin* is configured to a value greater than the value of *UnoccupiedSetback*, then the Thermostat server SHALL update the value of *UnoccupiedSetback* to equal the new value of *UnoccupiedSetbackMin*.

6.3.2.2.4.9 UnoccupiedSetbackMax Attribute

Specifies the maximum degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the *UnoccupiedSetback* attribute to be configured by a user.

A value of 0xff indicates the attribute is unused.

UnoccupiedSetbackMax attribute value SHALL be greater than *UnoccupiedSetbackMin* attribute value. Attempts to configure *UnoccupiedSetbackMax* with a value less than or equal to the value of the Uno*ccupiedSetbackMin* attribute SHALL cause the Thermostat server to respond with a Write Attribute Response Command containing the status INVALID_VALUE and SHALL revert back to the previous *UnoccupiedSetbackMax* attribute value.

If *UnoccupiedSetbackMax* is configured to a value less than the value of *UnoccupiedSetback*, then the Thermostat server SHALL update the value of *UnoccupiedSetback* to equal the new value of *UnoccupiedSetbackMax*.

6.3.2.2.4.10 EmergencyHeatDelta Attribute

Specifies the delta, in 0.1 degrees Celsius, between *LocalTemperature* and the *OccupiedHeatingSetpoint or UnoccupiedHeatingSetpoint* attributes at which the Thermostat server will operate in emergency heat mode.

If the difference between *LocalTemperature* and *Un/OccupiedHeatingSetpoint* is greater than or equal to the *EmergencyHeatDelta* and the Thermostat server's *SystemMode* attribute is in a heating-related mode, then the Thermostat server SHALL immediately switch to the *SystemMode* attribute value that provides the highest stage of heating (e.g., emergency heat) and continue operating in that running state until the *OccupiedHeatingSetpoint* value is reached. For example:

- *LocalTemperature* = 10.0 degrees Celsius
- *OccupiedHeatingSetpoint* = 16.0 degrees Celsius
- *EmergencyHeatDelta* = 2.0 degrees Celsius

=> *OccupiedHeatingSetpoint - LocalTemperature* ≥? *EmergencyHeatDelta*

=> 16 - 10 ≥? 2

=> TRUE >>> Thermostat server changes its SystemMode to operate in 2nd stage or emergency heat mode

The purpose of this attribute is to provide Thermostat clients the ability to configure rapid heating when a setpoint is of a specified amount greater than the measured temperature. This allows the heated space to be quickly heated to the desired level set by the user.

6.3.2.2.5 AC Information Attribute Set

Table 6-28. Attributes of the AC Information Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0040	*ACType*	enum8	0x00 – 0x04	RW	0x00	O
0x0041	*ACCapacity*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0042	*ACRefrigerantType*	enum8	0x00-0x03	RW	0x00	O
0x0043	*ACCompressorType*	enum8	0x00-0x03	RW	0x00	O
0x0044	*ACErrorCode*	map32	0x00000000 – 0xffffffff	RW	0x00000000	O
0x0045	*ACLouverPosition*	enum8	0x00 – 0x05	RW	0x00	O
0x0046	*ACCoilTemperature*	int16	0x954d – 0x7fff	R	-	O
0x0047	*ACCapacityFormat*	enum8	0x00 – 0xff	RW	0x00	O

6.3.2.2.5.1 *ACType* Attribute

Indicates the type of Mini Split *ACType* of Mini Split AC is defined depending on how Cooling and Heating condition is achieved by Mini Split AC.

Table 6-29. *ACType* Enumeration

Enumeration Field Value	Description
0x00	Reserved
0x01	Cooling and Fixed Speed
0x02	Heat Pump and Fixed Speed
0x03	Cooling and Inverter
0x04	Heat Pump and Inverter

6.3.2.2.5.2 *ACCapacity* Attribute

Indicates capacity of Mini Split AC in terms of the format defined by the *ACCapacityFormat* attribute

6.3.2.2.5.3 ACRefrigerantType Attribute

Indicates type of refrigerant used within the Mini Split AC.

Table 6-30. *ACRefrigerantType* Enumeration

Enumeration Field Value	Description
0x00	Reserved

Enumeration Field Value	Description
0x01	R22
0x02	R410a
0x03	R407c

6.3.2.2.5.4 ACCompressorType Attribute

This indicates type of Compressor used within the Mini Split AC.

Table 6-31. *ACCompressorType* Enumeration

Enumeration Field Value	Description
0x00	Reserved
0x01	T1, Max working ambient 43 ºC
0x02	T2, Max working ambient 35 ºC
0x03	T3, Max working ambient 52 ºC

6.3.2.2.5.5 ACErrorCode Attribute

This indicates the type of errors encountered within the Mini Split AC. Error values are reported with four bytes values. Each bit within the four bytes indicates the unique error.

Table 6-32. *ACErrorCode* Values

Bit	Value
0	Compressor Failure or Refrigerant Leakage
1	Room Temperature Sensor Failure
2	Outdoor Temperature Sensor Failure
3	Indoor Coil Temperature Sensor Failure
4	Fan Failure

6.3.2.2.5.6 ACLouverPosition Attribute

This indicates the position of Louver on the AC. Attribute values are listed in Table 6-33.

Table 6-33. ACLouverPosition Values

Louver Position Byte	Value
Fully Closed	0x01
Fully Open	0x02
Quarter Open	0x03
Half Open	0x04

Louver Position Byte	Value
Three Quarters Open	0x05

6.3.2.2.5.7 ACCoilTemperature Attribute

ACCoilTemperature represents the temperature in degrees Celsius, as measured locally or remotely (over the network) as follows:

- *ACCoilTemperature* = 100 x temperature in degrees Celsius.
- Where -273.15°C <= temperature <= 327.67 ºC, corresponding to an *ACCoilTemperature* in the range 0x954d to 0x7fff.
- The maximum resolution this format allows is 0.01 ºC.
- *ACCoilTemperature* of 0x8000 indicates that the temperature measurement is invalid.

6.3.2.2.5.8 ACCapacityFormat Attribute

This is the format for the *ACCapacity* attribute.

Table 6-34. ACCapacity Enumeration

Enumeration Field Value	Description
0x00	BTUh

6.3.2.3 Server Commands Received

The command IDs for the Thermostat cluster are listed in Table 6-35:

Table 6-35. Command IDs for the Thermostat Cluster

Command Identifier Field Value	Description	M/O
0x00	Setpoint Raise/Lower	M
0x01	Set Weekly Schedule	O
0x02	Get Weekly Schedule	O
0x03	Clear Weekly Schedule	O
0x04	Get Relay Status Log	O

6.3.2.3.1 Setpoint Raise/Lower Command

6.3.2.3.1.1 Payload Format

The Setpoint Raise/Lower command payload SHALL be formatted as illustrated in Figure 6-4Format of the Setpoint Raise/Lower Command Payload.

Figure 6-4. Format of the Setpoint Raise/Lower Command Payload

Bits	8	8
Data Type	enum8	int8
Field Name	Mode	Amount

6.3.2.3.1.2 Mode Field

The mode field SHALL be set to one of the non-reserved values in Table 6-36. It specifies which setpoint is to be configured. If it is set to auto, then both setpoints SHALL be adjusted.

Table 6-36. Mode Field Values for Setpoint Raise/Lower Command

Mode Field Value	Description
0x00	Heat (adjust Heat Setpoint)
0x01	Cool (adjust Cool Setpoint)
0x02	Both (adjust Heat Setpoint and Cool Setpoint)

6.3.2.3.1.3 Amount Field

The amount field is a signed 8-bit integer that specifies the amount the setpoint(s) are to be increased (or decreased) by, in steps of 0.1°C.

6.3.2.3.1.4 Effect on Receipt

The attributes for the indicated setpoint(s) SHALL be increased by the amount specified in the Amount field.

6.3.2.3.2 Set Weekly Schedule Command

6.3.2.3.2.1 Payload Format

The set weekly schedule command payload SHALL be formatted as shown in Figure 6-5 and Figure 6-6.

Figure 6-5. Set Weekly Schedule Command Payload Format (1 of 2)

Octets	1(Header)	1(Header)	1(Header)	2	2/0	2/0
Data Type	enum8	map8	map8	uint16	int16	int16
Field Name	Number of Transitions for Sequence	Day of Week for Sequence	Mode for Sequence	Transition Time 1	Heat Set Point 1	Cool Set Point 1

Figure 6-6. Set Weekly Schedule Command Payload Format (2 of 2)

Octets	Variable	2	2/0	2/0
Data Type	--	uint16	int16	int16
Field Name	--	Transition Time 10	Heat Set Point 10	Cool Set Point 10

The set weekly schedule command is used to update the thermostat weekly set point schedule from a management system. If the thermostat already has a weekly set point schedule programmed then it SHOULD replace each daily set point set as it receives the updates from the management system. For example if the thermostat has 4 set points for every day of the week and is sent a Set Weekly Schedule command with one set point for Saturday then the thermostat SHOULD remove all 4 set points for Saturday and replace those with the updated set point but leave all other days unchanged. If the schedule is larger than what fits in one frame or contains more than 10 transitions, the schedule SHALL then be sent using multiple Set Weekly Schedule Commands.

Each Set Weekly Schedule Command has 3 header bytes – Number of Transitions for Sequence, Day of Week for Sequence, and Mode for Sequence. The application SHALL decode the payload according to what has specified in the 3 header bytes.

6.3.2.3.2.2 Number of Transitions for Sequence

The Number of Transitions for Sequence field indicates how many individual transitions to expect for this sequence of commands. If a device supports more than 10 transitions in its schedule they can send this by sending more than 1 “Set Weekly Schedule” command, each containing the separate information that the device needs to set.

6.3.2.3.2.3 Day of Week for Sequence

This field represents the day of the week at which all the transitions within the payload of the command SHOULD be associated to. This field is a bitmap and therefore the associated set point could overlap onto multiple days (you could set one transition time for all “week days” or whatever combination of days the implementation requests). Table 6-37 displays the bitmap values.

Table 6-37. Day Of Week for Sequence Values

Bit Number	Description
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday
7	Away or Vacation

Each set point transition will begin with the day of week for this transition. There can be up to 10 transitions for each command.

6.3.2.3.2.4 Mode for Sequence

This field determines how the application SHALL decode the Set Point Fields of each transition for the remaining of the command. This field is a bitmap and the values are presented in Table 6-38.

Table 6-38. Mode for Sequence Values

Bit Number	Description
0	Heat Setpoint Field Present in Payload

Bit Number	Description
1	Cool Setpoint Field Present in Payload

If the Heat Bit is On and the Cool Bit is Off, the Command SHALL be represented as in Figure 6-7 and Figure 6-8.

Figure 6-7. Set Heat Weekly Schedule Command Payload Format (1 of 2)

Octets	1(Header)	1(Header)	1(Header)	2	2
Data Type	enum8	map8	map8	uint16	int16
Field Name	Number of Transitions for Sequence	Day of Week for Sequence	0x01 (Heat)	Transition Time 1	Heat Set Point 1

Figure 6-8. Set Heat Weekly Schedule Command Payload Format (2 of 2)

Octets	Variable	2	2
Data Type	--	uint16	int16
Field Name	--	Transition Time 10	Heat Set Point 10

If the Heat Bit is Off and the Cool Bit is On, the Command SHALL be represented as in Figure 6-9 and Figure 6-10.

Figure 6-9. Set Cool Weekly Schedule Command Payload Format (1 of 2)

Octets	1(Header)	1(Header)	1(Header)	2	2
Data Type	enum8	map8	map8	uint16	int16
Field Name	Number of Transitions for Sequence	Day of Week for Sequence	0x02 (Cool)	Transition Time 1	Cool Set Point 1

Figure 6-10. Set Cool Weekly Schedule Command Payload Format (2 of 2)

Octets	Variable	2	2
Data Type	--	uint16	int16
Field Name	--	Transition Time 10	Cool Set Point 10

If both the Heat Bit and the Cool Bit are On, the Command SHALL be represented as in Figure 6-11 and Figure 6-12.

Figure 6-11. Set Heat & Cool Weekly Schedule Command Payload Format (1 of 2)

Octets	1(Header)	1(Header)	1(Header)	2	2	2
Data Type	enum8	map8	map8	uint16	int16	int16

Field Name	Number of Transitions for Sequence	Day of Week for Sequence	0x03 (Heat & Cool)	Transition Time 1	Heat Set Point 1	Cool Set Point 1

Figure 6-12. Set Heat & Cool Weekly Schedule Command Payload Format (2 of 2)

Octets	Variable	2	2	2
Data Type	--	uint16	int16	int16
Field Name	--	Transition Time 10	Heat Set Point 10	Cool Set Point 10

At least one of the bits in the Mode For Sequence byte SHALL be on.

6.3.2.3.2.5 Transition Time Field

This field represents the start time of the schedule transition during the associated day. The time will be represented by a 16 bits unsigned integer to designate the minutes since midnight. For example, 6am will be represented by 0x0168 (360 minutes since midnight) and 11:30pm will be represented by 0x0582 (1410 minutes since midnight)

6.3.2.3.2.6 Heat Set Point Field

If the heat bit is enabled in the *Mode For Sequence* byte, this field represents the heat setpoint to be applied at this associated transition start time. The format of this attribute represents the temperature in degrees Celsius with 0.01 deg C resolution.

6.3.2.3.2.7 Cool Set Point Field

If the cool bit is enabled in the *Mode For Sequence* byte, this field represents the cool setpoint to be applied at this associated transition start time. The format of this attribute represents the temperature in degrees Celsius with 0.01 deg C resolution.

6.3.2.3.2.8 Effect on Receipt

The weekly schedule for updating set points SHALL be stored in the thermostat and SHOULD begin at the time of receipt. A default response SHALL always be sent as a response. If the total number of transitions sent is greater than what the thermostat supports a default response of INSUFFICIENT_SPACE (0x89) SHALL be sent in response to the last command sent for that transition sequence. If any of the set points sent in the entire sequence is out of range of what the thermostat supports (AbsMin/MaxSetPointLimit) then a default response of INVALID_VALUE (0x87) SHALL be sent in return and the no set points from the entire sequence SHOULD be used. If the transitions could be added successfully a default response of SUCCESS(0x00) SHALL be sent. Overlapping transitions is not allowed. If an overlap is detected and a default response of FAILURE(0x01) SHALL be sent. The Day of Week for Sequence and Mode for Sequence fields are defined as bitmask for the flexibility to support multiple days and multiple modes within one command. If thermostat cannot handle incoming command with multiple days and/or multiple modes within one command, it SHALL send default response of INVALID_FIELD (0x85) in return.

6.3.2.3.3 Get Weekly Schedule

Figure 6-13. Format of the Get Weekly Schedule Command Payload

Octets	1	1
Data Type	map8	map8
Field Name	Days To Return	Mode To Return

6.3.2.3.3.1 Days To Return

This field indicates the number of days the client would like to return the set point values for and could be any combination of single days or the entire week. This field has the same format as the Day of Week for Sequence field in the **Set Weekly Schedule command**.

6.3.2.3.3.2 Mode To Return

This field indicates the mode the client would like to return the set point values for and could be any combination of heat only, cool only or heat&cool. This field has the same format as the Mode for Sequence field in the **Set Weekly Schedule command**.

6.3.2.3.3.3 Effect on Receipt

When this command is received the unit SHOULD send in return the Get Weekly Schedule Response[112] command. The Days to Return and Mode to Return fields are defined as bitmask for the flexibility to support multiple days and multiple modes within one command. If thermostat cannot handle incoming command with multiple days and/or multiple modes within one command, it SHALL send default response of INVALID_FIELD (0x85) in return.

6.3.2.3.4 Clear Weekly Schedule

The Clear Weekly Schedule command is used to clear the weekly schedule. The Clear weekly schedule has no payload.

6.3.2.3.4.1 Effect on Receipt

When this command is received, all transitions currently stored SHALL be cleared and a default response of SUCCESS (0x00) SHALL be sent in response. There are no error responses to this command.

6.3.2.3.5 Get Relay Status Log

The Get Relay Status Log command is used to query the thermostat internal relay status log. This command has no payload.

The log storing order is First in First Out (FIFO) when the log is generated and stored into the Queue.

The first record in the log (i.e., the oldest) one, is the first to be replaced when there is a new record and there is no more space in the log. Thus, the newest record will overwrite the oldest one if there is no space left.

The log storing order is Last In First Out (LIFO) when the log is being retrieved from the Queue by a client device.

Once the "Get Relay Status Log Response" frame is sent by the Server, the "Unread Entries" attribute SHOULD be decremented to indicate the number of unread records that remain in the queue.

[112] CCB 2251 the command is Get Weekly Schedule Response (not Current Weekly Schedule)

If the "Unread Entries" attribute reaches zero and the Client sends a new "Get Relay Status Log Request", the Server MAY send one of the following items as a response:

i) resend the last Get Relay Status Log Response

or

ii) generate new log record at the time of request and send Get Relay Status Log Response with the new data

For both cases, the "Unread Entries" attribute will remain zero.

6.3.2.3.5.1 Effect on Receipt

When this command is received, the unit SHALL respond with Relay Status Log command if the relay status log feature is supported on the unit.

6.3.2.4 Server Commands Sent

Table 6-39 shows the command sent by the server (received by the client).

Table 6-39. Server Commands Send Command ID

Command Identifier Field Value	Description
0x00	Get Weekly Schedule Response
0x01	Get Relay Status Log Response

6.3.2.4.1 Get Weekly Schedule Response

This command has the same payload format as the Set Weekly Schedule. Please refer to the payload detail in Section Set Weekly Schedule Command, Set Weekly Schedule Command, in this chapter.

6.3.2.4.2 Get Relay Status Log Response

This command is sent from the thermostat cluster server in response to the Get Relay Status Log. After the Relay Status Entry is sent over the air to the requesting client, the specific entry will be cleared from the thermostat internal log.

6.3.2.4.2.1 Payload Format

The relay status log command payload SHALL be formatted as shown in Figure 6-14.

Figure 6-14. Format of the Relay Status Log Payload

Octets	2	2	2	1	2	2
Data Type	uint16	map8	int16	uint8	int16	uint16
Field Name	Time of Day	Relay Status	Local Temperature	Humidity in Percentage	Set Point	Unread Entries

6.3.2.4.2.2 Time of Day Field

Represents the sample time of the day, in minutes since midnight, when the relay status was captured for this associated log entry. For example, 6am will be represented by 0x0168 (360 minutes since midnight) and 11:30pm will be represented by 0x0582 (1410 minutes since midnight).

6.3.2.4.2.3 Relay Status Field

Presents the relay status for thermostat when the log is captured. Each bit represents one relay used by the thermostat. If the bit is on, the associated relay is on and active. Each thermostat manufacturer can create its own mapping between the bitmask and the associated relay.

6.3.2.4.2.4 Local Temperature Field

Presents the local temperature when the log is captured. The format of this attribute represents the temperature in degrees Celsius with 0.01 deg C resolution.

6.3.2.4.2.5 Humidity Field

This field presents the humidity as a percentage when the log was captured.

6.3.2.4.2.6 Setpoint Field

Presents the target setpoint temperature when the log is captured. The format of this attribute represents the temperature in degrees Celsius with 0.01 deg C resolution.

6.3.2.4.2.7 Unread Entries Field

This field presents the number of unread entries within the thermostat internal log system.

6.3.2.5 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes command and according to the minimum and maximum reporting interval and reportable change settings described in Chapter 2, Foundation and whenever they change. The following attributes SHALL be reported:

- *LocalTemperature*
- *PICoolingDemand*
- *PIHeatingDemand*

Other attributes MAY optionally be reported.

6.3.2.6 Scene Table Extensions

If the Scenes server cluster (see 3.7) is implemented, the following extension fields SHALL be added to the Scenes table in the given order, i.e., the attribute listed as 1 is added first:

1) *OccupiedCoolingSetpoint*

2) *OccupiedHeatingSetpoint*

3) *SystemMode*

6.3.3 Client

The client has no specific dependencies nor specific attributes. The client cluster generates the commands received by the server cluster, as required by the application.

6.4 Fan Control

6.4.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster specifies an interface to control the speed of a fan as part of a heating / cooling system.

6.4.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added

6.4.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	FAN	Type 1 (client to server)

6.4.1.3 Cluster Identifiers

Identifier	Name
0x0202	Fan Control

6.4.2 Server

6.4.2.1 Attributes

The Fan Control Status attribute set contains the attributes summarized in Table 6-40Attributes of the Fan Control Cluster.

Table 6-40. Attributes of the Fan Control Cluster

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*FanMode*	enum8	0x00 – 0x06	RW	0x05 (auto)	M
0x0001	*FanModeSequence*	enum8	0x00 – 0x04	RW	0x02	M

6.4.2.1.1 *FanMode* Attribute

The *FanMode* attribute is an 8-bit value that specifies the current speed of the fan. It SHALL be set to one of the nonreserved values in Table 6-41:

Table 6-41. *FanMode* Attribute Values

Value	Description
0x00	Off
0x01	Low
0x02	Medium
0x03	High
0x04	On
0x05	Auto (the fan speed is self-regulated)
0x06	Smart (when the heated/cooled space is occupied, the fan is always on)

Note that for Smart mode, information must be available as to whether the heated/cooled space is occupied. This MAY be accomplished by use of the Occupancy Sensing cluster (see 4.8).

6.4.2.1.2 *FanModeSequence* Attribute

The *FanModeSequence* attribute is an 8-bit value that specifies the possible fan speeds that the thermostat can set. It SHALL be set to one of the non-reserved values in Table 6-42*FanSequenceOperatio*. (**Note:** 'Smart' is not in this table, as this mode resolves to one of the other modes depending on occupancy).

Table 6-42. *FanSequenceOperation* Attribute Values

Attribute Value	Description
0x00	Low/Med/High
0x01	Low/High
0x02	Low/Med/High/Auto
0x03	Low/High/Auto
0x04	On/Auto

6.4.2.2 Commands

No cluster specific commands are generated or received by the server.

6.4.3 Client

The Client cluster has no specific attributes. No cluster specific commands are received by the server. No cluster specific commands are generated by the server.

6.5 Dehumidification Control

6.5.1 Overview

This cluster provides an interface to dehumidification functionality.

6.5.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added

6.5.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	DHUM	Type 1 (client to server)

6.5.1.3 Cluster Identifiers

Identifier	Name
0x0203	Dehumidification Control

6.5.2 Server

6.5.2.1 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant nibble specifies the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute set for the dehumidification control cluster is listed in Table 6-43.

Table 6-43. Dehumidification Control Attribute Sets

Attribute Set Identifier	Description
0x000	Dehumidification Information
0x001	Dehumidification Settings

6.5.2.1.1 Dehumidification Information Attribute Set

The Dehumidification Information attribute set contains the attributes summarized in Table 6-44Dehumidification Information Attribute Set.

Table 6-44. Dehumidification Information Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*RelativeHumidity*	uint8	0x00 – 0x64	R	-	O
0x0001	*DehumidificationCooling*	uint8	0 - *DehumidificationMaxCool*	RP	-	M

6.5.2.1.1.1 *RelativeHumidity* Attribute

The *RelativeHumidity* attribute is an 8-bit value that represents the current relative humidity (in %) measured by a local or remote sensor. The valid range ix 0x00 – 0x64 (0% to 100%).

6.5.2.1.1.2 *DehumidificationCooling* Attribute

The *DehumidificationCooling* attribute is an 8-bit value that specifies the current dehumidification cooling output (in %). The valid range is 0 to *DehumidificationMaxCool*.

6.5.2.1.2 Dehumidification Settings Attribute Set

The Dehumidification Settings attribute set contains the attributes summarized in the table below:

Table 6-45. Dehumidification Settings Attribute Set

Identifier	Name	Type	Range	Access	Default	M/O
0x0010	*RHDehumidificationSetpoint*	uint8	0x1E – 0x64	RW	0x32	M
0x0011	*RelativeHumidityMode*	enum8	0x00 – 0x01	RW	0x00	O
0x0012	*DehumidificationLockout*	enum8	0x00 – 0x01	RW	0x01	O
0x0013	*DehumidificationHysteresis*	uint8	0x02 – 0x14	RW	0x02	M
0x0014	*DehumidificationMaxCool*	uint8	0x14 – 0x64	RW	0x14	M
0x0015	*RelativeHumidityDisplay*	enum8	0x00 – 0x01	RW	0x00	O

6.5.2.1.2.1 *RHDehumidificationSetpoint* Attribute

The *RHDehumidificationSetpoint* attribute is an 8-bit value that represents the relative humidity (in %) at which dehumidification occurs. The valid range ix 0x1E – 0x64 (30% to 100%).

6.5.2.1.2.2 *RelativeHumidityMode* Attribute

The *RelativeHumidityMode* attribute is an 8-bit value that specifies how the *RelativeHumidity* value is being updated. It SHALL be set to one of the values below:

Table 6-46. *RelativeHumidityMode* Attribute Values

Attribute Value	Description
0x00	*RelativeHumidity* measured locally
0x01	*RelativeHumidity* updated over the network

6.5.2.1.2.3 *DehumidificationLockout* Attribute

The *DehumidificationLockout* attribute is an 8-bit value that specifies whether dehumidification is allowed or not. It SHALL be set to one of the values below:

Table 6-47. *DehumidificationLockout* Attribute Values

Attribute Value	Description
0x00	Dehumidification is not allowed.
0x01	Dehumidification is allowed.

6.5.2.1.2.4 *DehumidificationHysteresis* Attribute

The *DehumidificationHysteresis* attribute is an 8-bit value that specifies the hysteresis (in %) associated with *RelativeHumidity* value. The valid range ix 0x02 – 0x14 (2% to 20%).

6.5.2.1.2.5 *DehumidificationMaxCool* Attribute

The *DehumidificationMaxCool* attribute is an 8-bit value that specifies the maximum dehumidification cooling output (in %). The valid range ix 0x14 – 0x64 (20% to 100%).

6.5.2.1.2.6 *RelativeHumidityDisplay* Attribute

The *RelativeHumidityDisplay* attribute is an 8-bit value that specifies whether the *RelativeHumidity* value is displayed to the user or not. It SHALL be set to one of the non-reserved values in Table 6-48.

Table 6-48. *RelativeHumidityMode* Attribute Values

Attribute Value	Description
0x00	*RelativeHumidity* is not displayed
0x01	*RelativeHumidity* is displayed

6.5.2.2 Commands

No cluster specific commands are generated or received by the server.

6.5.2.3 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes command and according to the minimum and maximum reporting interval settings described in the ZCL Foundation specification.

The following attribute SHALL be reported: *DehumidificationCooling*

This attribute SHALL also be reported whenever it changes (a minimum change is 1%).

Reports of this attribute MAY be used to control a remote dehumidifier device.

6.5.3 Client

The client has no dependencies or attributes and there are no cluster specific commands defined.

6.6 Thermostat User Interface Configuration

6.6.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface to allow configuration of the user interface for a thermostat, or a thermostat controller device, that supports a keypad and LCD screen.

6.6.1.1 Revision History

Rev	Description
1	global mandatory *ClusterRevision* attribute added

6.6.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	TSUIC	Type 1 (client to server)

6.6.1.3 Cluster Identifiers

Identifier	Name
0x0204	Thermostat User Interface Configuration

6.6.2 Server

6.6.2.1 Attributes

The attributes of this cluster are summarized in Table 6-49.

Table 6-49. Thermostat User Interface Configuration Cluster

Id	Name	Type	Range	Access	Default	M/O
0x0000	*TemperatureDisplayMode*	enum8	0x00 – 0x01	RW	0x00 (Celsius)	M
0x0001	*KeypadLockout*	enum8	0x00 – 0x05	RW	0x00 (no lockout)	M
0x0002	*ScheduleProgrammingVisibility*	enum8	0x00 – 0x01	RW	0x00	O

6.6.2.1.1 *TemperatureDisplayMode* Attribute

The *TemperatureDisplayMode* attribute specifies the units of the temperature displayed on the thermostat screen. This attribute SHALL be set to one of the non-reserved values in Table 6-50.

Table 6-50. *DisplayMode* Attribute Values

Attribute Value	Description
0x00	Temperature in ºC
0x01	Temperature in ºF

6.6.2.1.2 *KeypadLockout* Attribute

The *KeypadLockout* attribute specifies the level of functionality that is available to the user via the keypad. This attribute SHALL be set to one of the non-reserved values Table 6-51*KeypadLockou*.

Table 6-51. *KeypadLockout* Attribute Values

Attribute Value	Description
0x00	No lockout
0x01	Level 1 lockout
0x02	Level 2 lockout
0x03	Level 3 lockout
0x04	Level 4 lockout
0x05	Level 5 lockout (least functionality available to the user)

The interpretation of the various levels is device-dependent.

6.6.2.1.3 ScheduleProgrammingVisibility Attribute

The *ScheduleProgrammingVisibility* attribute is used to hide the weekly schedule programming functionality or menu on a thermostat from a user to prevent local user programming of the weekly schedule. The schedule programming MAY still be performed via a remote interface, and the thermostat MAY operate in schedule programming mode.

This command is designed to prevent local tampering with or disabling of schedules that MAY have been programmed by users or service providers via a more capable remote interface. The programming schedule SHALL continue to run even though it is not visible to the user locally at the thermostat.

It SHALL be set to one of the non-reserved values in Table 6-52.

Table 6-52. *ScheduleProgrammingVisibility* Attribute Values

ScheduleProgrammingVisibility Attribute Value	Description
0x00	Local schedule programming functionality is enabled at the thermostat
0x01	Local schedule programming functionality is disabled at the thermostat

6.6.2.2 Commands

No cluster specific commands are generated or received by the server.

6.6.2.3 Sample Conversion Code

Sample code provided to ensure consistent Fahrenheit to Celsius and vice-versa conversion between devices and across vendors.

```
For degF: the value is a int8u representing 2x temperature
value in Farenheit (to get 0.5 resolution).
For degC: the value is a int16s representing Celsius in
0.01 resolution as expected by the ZCL format.
/*
 * Function   :  translateZclTemp()
 * Description   :  Converts the temperature setpoints in ZCL
 *     to the half degF format.
 *     The half degF format is 8-bit unsigned,
 *     and represents 2x temperature value in
 *     Farenheit (to get 0.5 resolution).
 *     The format used in ZCL is 16-bit signed
 *     in Celsius and multiplied by 100
 *     to get 0.01 resolution.
 *     e.g. 2500(25.00 deg C) ---> 0x9A (77 deg F)
 * Input Para : Temperature in ZCL (degC)format
 * Output Para: Temperature in half DegF format
 */
int8u translateZclTemp(int16s temperature)
{
  int32s x = temperature;
  //rearrangement of
  // = (x * (9/5) / 100 + 32) * 2;
  // the added 250 is for proper rounding.
  // a rounding technique that only works
  // with positive numbers

  return (int8u) ((x*9*2 + 250)/ (5*100) + 64);
}

/*
 * Function   :  translateDegFTemp
 * Description   :  Converts the temperature in DegF
 *  protocol to the format
 *  expected by the cluster attribute
 *  Measured Value in the
 *  Temperature Measurement
 *  Information Attribute Set.
 *  The half deg F format is 8-bit
 *  unsigned, and represents
 *  2x temperature value in
 *  Farenheit (to get 0.5 resolution).
 *  The format expected by cluster
 *  is 16-bit signed in Celsius and
 *  multiplied by 100 to get
 *  0.01 resolution.
 *  e.g. 0x9A(77 deg F) ---> 2500 (25.00 deg C)
 * Input Para : temperature in DegF format
 * Output Para: temperature in ZCL format
 */
int16s translateDegFTemp(int8u temperature)
{
  int32s x = temperature;

  // rearrangement of
  // = 100 * (x/2 - 32) * 5/9
  // *1000 (should be 100), +90, then /10,
  // is for rounding.

  return (int16s) (((x - 64)*5*1000 + 90) / (10*2*9));
}
```

6.6.3 Client

The client has no dependencies or cluster specific attributes and there are no cluster specific commands defined.

CHAPTER 7 CLOSURES

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

7.1 General Description

7.1.1 Introduction

The clusters specified in this document are for use typically in applications involving closures (e.g., shades, windows, doors), but MAY be used in any application domain.

7.1.2 Cluster List

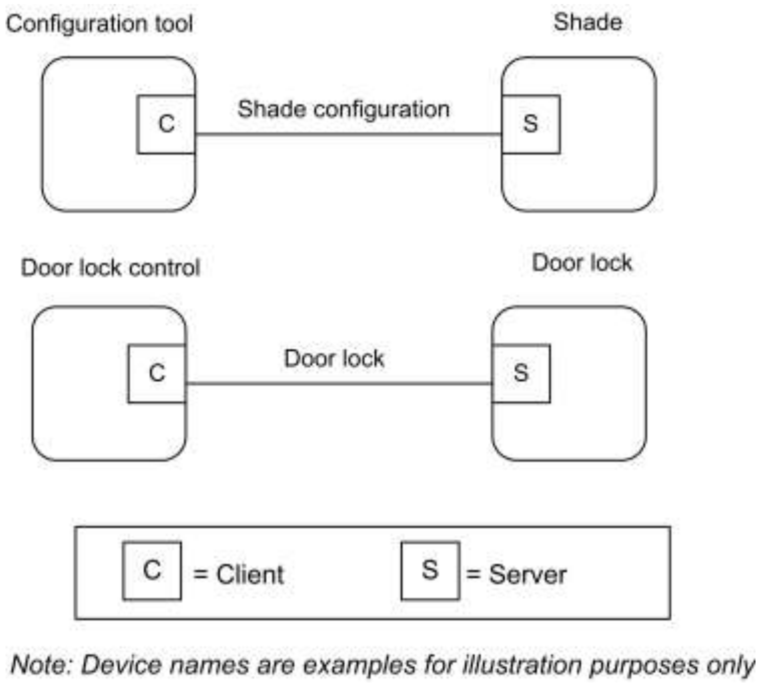
This section lists the clusters specified in this document, and gives examples of typical usage for the purpose of clarification.

The clusters defined in this document are listed in Table 7-1.

Table 7-1. Clusters Specified in the Closures Functional Domain

Cluster ID	Cluster Name	Description
0x0100	Shade Configuration	Attributes and commands for configuring a shade
0x0101	Door Lock	An interface to a generic way to secure a door
0x0102	Window Covering	Commands and attributes for controlling a window covering

Figure 7-1. Typical Usage of the Closures Clusters



7.2 Shade Configuration

7.2.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface for reading information about a shade, and configuring its open and closed limits.

7.2.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

7.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	SHDCFG	Type 2 (server to client)

7.2.1.3 Cluster Identifiers

Identifier	Name
0x0100	Shade Configuration

7.2.2 Server

7.2.2.1 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 7-2.

Table 7-2. Shade Configuration Attribute Sets

Attribute Set Identifier	Description
0x000	Shade information
0x001	Shade settings

7.2.2.1.1 Shade Information Attribute Set

The Shade Information attribute set contains the attributes summarized in Table 7-3.

Table 7-3. Attributes of the Shade Information Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0000	*PhysicalClosedLimit*	uint16	0x0001 – 0xfffe	R	-	O
0x0001	*MotorStepSize*	uint8	0x00 – 0xfe	R	-	O
0x0002	*Status*	map8	0000 xxxx	RW	0000 0000	M

7.2.2.1.1.1 *PhysicalClosedLimit* Attribute

The *PhysicalClosedLimit* attribute indicates the most closed (numerically lowest) position that the shade can physically move to. This position is measured in terms of steps of the motor, taking the physical most open position of the shade as zero.

This attribute is for installation informational purposes only.

The value 0xffff indicates an invalid or unknown *PhysicalClosedLimit*.

7.2.2.1.1.2 *MotorStepSize* Attribute

The *MotorStepSize* attribute indicates the angle the shade motor moves for one step, measured in 1/10ths of a degree.

This attribute is for installation informational purposes only.

The value 0xff indicates an invalid or unknown step size.

7.2.2.1.1.3 *Status* Attribute

The *Status* attribute indicates the status of a number of shade functions, as shown in Table 7-4. Writing a value to this attribute only affects those bits with Read/Write access.

Table 7-4. Bit Values for the *Status* Attribute

Bit Number	Meaning	Access
0	Shade operational 0 = no 1 = yes	R
1	Shade adjusting 0 = no 1 = yes	R
2	Shade direction 0 = closing 1 = opening	R
3	Direction corresponding to forward direction of motor 0 = closing 1 = opening	RW

7.2.2.1.2 Shade Settings Attribute Set

The Shade Settings attribute set contains the attributes summarized in Table 7-5.

Table 7-5. Attributes of the Shade Settings Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0010	*ClosedLimit*	uint16	0x0001 – 0xfffe	RW	0x0001	M
0x0011	*Mode*	enum8	0x00 – 0xfe	RW	0x00	M

7.2.2.1.2.1 *ClosedLimit* Attribute

The *ClosedLimit* attribute indicates the most closed position that the shade can move to. This position is measured in terms of steps of the motor, taking the physical most open position of the shade as zero. This attribute is set either by directly writing it, or by the following method.

When the Mode attribute is set to Configure, the shade is opening, and either the shade is stopped or it reaches its physical most open limit (if there is one – the motor MAY continue to turn at the top), the zero point for the motor-step measurement system is set to the current position of the shade.

When the Mode attribute is set to Configure, the shade is closing, and either the shade is stopped or it reaches its physical closed limit, the *ClosedLimit* attribute is set to the current position of the shade, relative to the zero point set as described above.

7.2.2.1.2.2 *Mode* Attribute

The *Mode* attribute indicates the current operating mode of the shade, as shown in Table 7-6.The value 0xff indicates an invalid or unknown mode.

Table 7-6. Values of the Mode **Attribute**

Attribute Value	Meaning
0x00	Normal
0x01	Configure

In configure mode, the *ClosedLimit* attribute MAY be set as described above.

7.2.2.2 Commands

No cluster specific commands are generated or received by the server.

7.2.3 Client

The client has no specific attributes and there are no cluster specific commands to receive or generate.

7.3 Door Lock

7.3.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The door lock cluster provides an interface to a generic way to secure a door. The physical object that provides the locking functionality is abstracted from the cluster. The cluster has a small list of mandatory attributes and functions and a list of optional features.

7.3.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added; CCB 1811 1812 1821
2	CCB 2430

7.3.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	DRLK	Type 2 (server to client)

7.3.1.3 Cluster Identifiers

Identifier	Name
0x0101	Door Lock

7.3.2 Server

Generally the door lock itself implements the server side of this cluster. The attributes and commands listed in this cluster were developed to be implemented by a door lock which has the ability to keep track of multiple users and schedules.

7.3.2.1 Alarms, Reports, and Events

A door lock implementing all of the optional features provided in this cluster has the ability to push data to a controller in three different forms, Alarms, Reports and Events. Alarms are used to report critical states on the door lock. Reports are used to inform a subscribed device of changes of state in specific attributes on the lock. Events are used to inform a bound device about changes in state related to the operation and programming of the door lock. Event commands are sent to a binding. Examples of events are locking and unlocking the lock and adding or deleting a user on the lock.

7.3.2.1.1 Alarms

The door lock cluster provides several alarms which can be sent when there is a critical state on the door lock. The alarms available for the door lock cluster are listed in the section below outlining the alarm mask attribute. The Alarm cluster is used to generate the actual alarms.

Alarm example: If the first bit of the attribute Alarm Mask is set, any device that is bound to the alarm cluster will be informed each time the deadbolt becomes jammed. If for some reason the door lock became jammed, the door lock would send an alarm command from the alarms cluster with the payload illustrated in Figure 7-2.

Figure 7-2. Format of the Alarm Cluster

Octets	1	2
Data Type	enum8	Cluster ID
Field Name	Alarm Code	Cluster Identifier
Field Value	0x00	0x0101

7.3.2.1.2 Reports

The reporting mechanism within the ZCL can be used to subscribe to changes in a specific attribute within the door lock.

Report example: If an application wants to know each time a programming change is made on the door lock, it can use the reporting mechanism to be informed of changes to the Operating Mode attribute. Each time the Operating Mode changes to programming mode, the application will be informed and can then sync its knowledge of user data to make sure that it has an up to date record of user the users supported on the door lock.

7.3.2.1.3 Events

The event mechanism described within this document is unique to the Door Lock Cluster and was designed specifically for this cluster and no other. It is in part modeled on similar mechanisms in other clusters such as the load control events in the Demand Response and Load Control cluster in Smart Energy (DRLC).

The event mechanism in the door lock centers on the transmission of two commands autonomously generated by the server and sent to a bound device. The assumption is that the binding mechanism will be used to commission the server to send these commands.

There are two types of events on the door lock, operational and programmatic. Operational events relate to the general operation of the door lock, when it locks and unlocks for instance. The programmatic events relate to the programming of the door lock, for example when users are added or modified via the keypad.

Events are transmitted using two server commands, the Operation Event Notification Command and Programming Event Notification Command.

A primary key uniquely identifies each event. The key consists of the event's type (operation, programming etc…), source (keypad, RF, manual, etc…) and event code. The event mask bit that matches its type, source and event code controls the generation of each event. A complete list of events is included in the description of their commands along with the specific attribute and bit that control their generation.

7.3.2.2 Door Lock Security

The following functionality has not been validated at a Specification Validation Event and is therefore considered provisional.

Door locks have the ability to require the use of APS encryption for sending and receiving of all cluster messages. The Security Level attribute is used to specify the type of encryption required by the door lock.

The APS key MUST be unique to the door lock device to provide the enhanced security needed. Therefore, if APS security setting is selected, the device SHALL use a randomly generated install code to generate the unique APS link key to join to the network and use this unique APS link key to encryption all Door Lock Cluster, Group Cluster, Scene Cluster messages.

The hashing method used to convert install code into APS link key is AES-MMO.

It SHOULD be noted that for the device with unique APS link key to join successfully to the network, the Trust Center will need to have a method for the user/installer to input the unique install code for the device.

Note that the security setting will only take effect when the device is not part of a network. If the user modifies the Security Level setting while the device is part of a network, the setting will not be applied until the device leaves the network and commissions to a network again.

7.3.2.3 Time

There are various references to the LocalTime within this cluster specification.

LocalTime (32-bit unsigned integer) represents the number of seconds since January 1 2000, in the local zone with time saving adjusted.

7.3.2.4 PIN/RFID Code Format

The PIN/RFID codes defined in this specification are all in ZCL OCTET STRING format. The first octet in the string specifies the number of octets contained in the remaining of the data field not including itself.

All value in the PIN/RFID code SHALL be ASCII encoded regardless if the PIN/RFID codes are number or characters. For example, code of “1, 2, 3, 4” SHALL be represented as 0x31, 0x32, 0x33, 0x34.

7.3.2.5 Process for Creating a New User with Schedule

The following is the process that the client device SHALL follow for creating a new user with weekday schedule or yearday schedule. The following process SHOULD be implemented as an atomic set and SHOULD not be broken up.

1. Set Pin Code
2. Set Weekday Schedule or Set Yearday Schedule
3. Set User Type to the desired schedule user type.

7.3.2.6 Process for Clearing All Schedules for a User

The following is the process that the client device SHALL follow for clearing all weekday schedule or all yearday schedule for a user. The following process SHOULD be implemented as an atomic set and SHOULD not be broken up.

1. Clear All Weekday Schedule or Clear All Yearday Schedule
2. Set User Type to the Unrestricted User Type

Note: If the User Type is not reset to Unrestricted User, the associated user Code (ex: PIN/RFID) will not have access.

7.3.2.7 Clarification of Changing the User Type

When the user type is changed from a scheduled user to some other user type, the door lock server MAY remove the user's schedule.

7.3.2.8 Clarification for Changing the User Code

When changing the user code, the server SHALL not require that the user code be removed first.

7.3.2.9 Server Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets for Door Lock Cluster Server are listed in Table 7-7.

Table 7-7. Attribute Sets Description

Attribute Set	Identifier Description
0x0000 – 0x000F	Basic Information Attribute Set
0x0010 – 0x001F	User, PIN, Schedule Information Attribute Set
0x0020 – 0x002F	Operational Settings Attribute Set
0x0030 – 0x003F	Security Settings Attribute Set
0x0040 – 0x004F	Alarm and Event Masks Attribute Set

7.3.2.10 Basic Information Attribute Set

Table 7-8. Current Information Attribute Set

Identifier	Name	Type	Access	Def	M/O
0x0000	*LockState*	enum8	RP	-	M
0x0001	*LockType*	enum8	R	-	M
0x0002	*ActuatorEnabled*	bool	R	-	M
0x0003	*DoorState*	enum8	RP	-	O
0x0004	*DoorOpenEvents*	uint32	RW	-	O
0x0005	*DoorClosedEvents*	uint32	RW	-	O
0x006	*OpenPeriod*	uint16	RW	-	O

7.3.2.10.1 *LockState* Attribute

This attribute has the following possible values:

Table 7-9. *LockState* Attribute Values

Value	Definition
0x00	Not fully locked

Value	Definition
0x01	Locked
0x02	Unlocked
0xFF	Undefined

7.3.2.10.2 *LockType* Attribute

The *LockType* attribute is indicated by an enumeration:

Table 7-10. *LockType* Attribute Values

Value	Definition
0x00	Dead bolt
0x01	Magnetic
0x02	Other
0x03	Mortise
0x04	Rim
0x05	Latch Bolt
0x06	Cylindrical Lock
0x07	Tubular Lock
0x08	Interconnected Lock
0x09	Dead Latch
0x0A	Door Furniture

7.3.2.10.3 ActuatorEnabled Attribute

This attribute has the following possible values:

Table 7-11. *ActuatorEnabled* Attribute Values

Boolean Value	Definition
0	Disabled
1	Enabled

7.3.2.10.4 *DoorState* Attribute

This attribute has the following possible values:

Table 7-12. *DoorState* Attribute Values

Value	Definition
0x00	Open
0x01	Closed
0x02	Error (jammed)
0x03	Error (forced open)
0x04	Error (unspecified)
0xFF	Undefined

7.3.2.10.5 DoorOpenEvents Attribute

This attribute holds the number of door open events that have occurred since it was last zeroed.

7.3.2.10.6 DoorClosedEvents Attribute

This attribute holds the number of door closed events that have occurred since it was last zeroed.

7.3.2.10.7 *OpenPeriod* Attribute

This attribute holds the number of minutes the door has been open since the last time it transitioned from closed to open.

7.3.2.11 User, PIN, Schedule, Log Information Attribute Set

Table 7-13. User, PIN, Schedule, Log Information Attribute Set

Id	Description	Type	Access	Def	M/O
0x0010	*NumberOfLogRecordsSupported*	uint16	R	0	O
0x0011	*NumberOfTotalUsersSupported*	uint16	R	0	O
0x0012	*NumberOfPINUsersSupported*	uint16	R	0	O
0x0013	*NumberOfRFIDUsersSupported*	uint16	R	0	O
0x0014	*NumberOfWeekDaySchedulesSupportedPerUser*	uint8	R	0	O
0x0015	*NumberOfYearDaySchedulesSupportedPerUser*	uint8	R	0	O
0x0016	*NumberOfHolidaySchedulesSupported*	uint8	R	0	O

Id	Description	Type	Access	Def	M/O
0x0017	*MaxPINCodeLength*	uint8	R	0x08	O
0x0018	*MinPINCodeLength*	uint8	R	0x04	O
0x0019	*MaxRFIDCodeLength*	uint8	R	0x14	O
0x001A	*MinRFIDCodeLength*	uint8	R	0x08	O

7.3.2.11.1 NumberOfLogRecordsSupported Attribute

The number of available log records.

7.3.2.11.2 NumberOfTotalUsersSupported Attribute

Number of total users supported by the lock. This value is equal to the higher one of [# of PIN Users Supported] and [# of RFID Users Supported]

7.3.2.11.3 NumberOfPINUsersSupported Attribute

The number of PIN users supported.

7.3.2.11.4 NumberOfRFIDUsersSupported Attribute

The number of RFID users supported.

7.3.2.11.5 NumberOfWeekDaySchedulesSupportedPerUser Attribute

The number of configurable week day schedule supported per user.

7.3.2.11.6 NumberOfYearDaySchedulesSupportedPerUser Attribute

The number of configurable year day schedule supported per user.

7.3.2.11.7 NumberOfHolidaySchedulesSupported Attribute

The number of holiday schedules supported for the entire door lock device.

7.3.2.11.8 MaxPINCodeLength Attribute

An 8 bit value indicates the maximum length in bytes of a PIN Code on this device. The default is set to 8 since most lock manufacturers currently allow PIN Codes of 8 bytes or less.

7.3.2.11.9 MinPINCodeLength Attribute

An 8 bit value indicates the minimum length in bytes of a PIN Code on this device. The default is set to 4 since most lock manufacturers do not support PIN Codes that are shorter than 4 bytes.

7.3.2.11.10 MaxRFIDCodeLength Attribute

An 8 bit value indicates the maximum length in bytes of a RFID Code on this device. The value depends on the RFID code range specified by the manufacturer, if media anti-collision identifiers (UID) are used as RFID code, a value of 20 (equals 10 Byte ISO 14443A UID) is recommended.

7.3.2.11.11 MinRFIDCodeLength Attribute

An 8 bit value indicates the minimum length in bytes of a RFID Code on this device. The value depends on the RFID code range specified by the manufacturer, if media anti-collision identifiers (UID) are used as RFID code, a value of 8 (equals 4 Byte ISO 14443A UID) is recommended.

7.3.2.12 Operational Settings Attribute Set

The attributes within this attribute set affect the physical behavior on the server device. Some of the setting might not be applicable to the specific device. When the client sends the write attribute request with values that are not applicable to the server device, the server SHALL send back a Write Attribute Response with error status not equal to ZCL_SUCCESS(0x00). It is suggested that it SHOULD respond with an error status of ZCL_INVALID_VALUE (0x87).

Table 7-14. Operational Settings Attribute Set

Id	Description	Type	Access	Def	M/O
0x0020	*EnableLogging*	bool	R*W P	0	O
0x0021	*Language*	string (3 bytes)	R*W P	0	O
0x0022	*LEDSettings*	uint8	R*W P	0	O
0x0023	*AutoRelockTime*	uint32	R*W P	0	O
0x0024	*SoundVolume*	uint8	R*W P	0	O
0x0025	*OperatingMode*	enum8	R*W P	0	O
0x0026	*SupportedOperatingModes*	map16	R	0x0001	O
0x0027	*DefaultConfigurationRegister*	map16	RP	0x0000	O
0x0028	*EnableLocalProgramming*	bool	R*W P	0x01	O
0x0029	*EnableOneTouchLocking*	bool	RWP	0	O
0x002A	*EnableInsideStatusLED*	bool	RWP	0	O
0x002B	*EnablePrivacyModeButton*	bool	RWP	0	O

7.3.2.12.1 EnableLogging Attribute

Enable/disable event logging. When event logging is enabled, all event messages are stored on the lock for retrieval. Logging events can be but not limited to Tamper Alarm, Lock, Unlock, Autolock, User Code Added, User Code Deleted, Schedule Added, and Schedule Deleted. For a full detail of all the possible alarms and events, please refer to the full list in the Alarm and Event Masks Attribute Set.

7.3.2.12.2 *Language* Attribute

Modifies the language for the on-screen or audible user interface using three bytes from ISO-639-1. It consists of one byte of length and two bytes for the language code. For example if the language is set to English, the value would be "02 65 6E" for the language code "en".

7.3.2.12.3 OperatingMode Attribute

Table 7-15 shows the current operating mode and which interfaces are enabled during each of the operating mode.

Table 7-15. Operating Modes

Enum	Operating Mode	Interface (E = Enabled; D = Disabled)		
		Keypad	RF	RFID
0x00	Normal	E	E	E
0x01	Vacation	D	E	E
0x02	Privacy	D	D	D
0x03	No RF Lock/Unlock	E	D	E
0x04	Passage	N/A	N/A	N/A

Normal Mode: The lock operates normally. All interfaces are enabled.

Vacation Mode: Only RF interaction is enabled. The keypad cannot be operated.

Privacy Mode: All external interaction with the door lock is disabled. This is intended so that users presumably inside the property will have control over the entrance. Privacy mode assumes that the lock can only be operated from inside by operating the thumb turn or some other means of ending privacy mode.

No RF Lock or Unlock: This mode only disables RF interaction with the lock. It specifically applies to the Lock, Unlock, Toggle, and Unlock with Timeout Commands.

Passage Mode: The lock is open or can be open or closed at will without the use of a Keypad or other means of user validation.

Note: For modes that disable the RF interface, the door lock SHALL respond to Lock, Unlock, Toggle, and Unlock with Timeout commands with a ZCL response with status FAILURE (0x01) and not take the action requested by those commands. The door lock SHALL NOT disable the radio or otherwise unbind or leave the network. It SHALL still respond to all other commands and requests.

7.3.2.12.4 SupportedOperatingModes Attribute

This bitmap contains all operating bits of the Operating Mode Attribute supported by the lock. The value of the enumeration in “Operating Mode” defines the related bit to be set, as shown in Table 7-16. All bits supported by a lock SHALL be set to zero.

Table 7-16. Bit Values for the *SupportedOperatingModes* Attribute

Bitmap Number	Description
0	Normal Mode Supported
1	Vacation Mode Supported
2	Privacy Mode Supported
3	No RF Lock or Unlock Mode Supported
4	Passage Mode Supported

7.3.2.12.5 LEDSettings Attribute

The settings for the LED support three different modes, shown in Table 7-17:

Table 7-17. Modes for the *LEDSettings* Attribute

Attribute Identifier	Definition
0x00	Never use LED for signalization
0x01	Use LED signalization except for access allowed events
0x02	Use LED signalization for all events

7.3.2.12.6 AutoRelockTime Attribute

The number of seconds to wait after unlocking a lock before it automatically locks again. 0=disabled. If set, unlock operations from any source will be timed. For one time unlock with timeout use the specific command.

7.3.2.12.7 SoundVolume Attribute

The sound volume on a door lock has three possible settings: silent, low and high volumes, shown in Table 7-18.

Table 7-18. Settings for the *SoundVolume* Attribute

Attribute Identifier	Definition
0x00	Silent Mode
0x01	Low Volume
0x02	High Volume

7.3.2.12.8 DefaultConfigurationRegister Attribute

This attribute represents the default configurations as they are physically set on the device (example: hardware dip switch setting, etc…) and represents the default setting for some of the attributes within this Operational Setting Attribute Set (for example: LED, Auto Lock, Sound Volume, and Operating Mode attributes), as in Table 7-19.

This is a read-only attribute and is intended to allow clients to determine what changes MAY need to be made without having to query all the included attributes. It MAY be beneficial for the clients to know what the device's original settings were in the event that the device needs to be restored to factory default settings.

If the Client device would like to query and modify the door lock server's operating settings, it SHOULD send read and write attribute request to the specific attributes.

For example, the Buzzer bitmap within this attribute is off. It represents the hardware dip switch Buzzer setting (original default setting) is off and the Sound Volume attribute default value is in Silent Mode. However, it is possible that the current Sound Volume is in High Volume. Therefore, if the client wants to query/modify the current Sound Volume setting on the server, the client SHOULD read/write to the Sound Volume attribute.

Table 7-19. *DefaultConfigurationRegister* Attribute

Bitmap Number	Description
0	0 - Enable Local Programming Attribute default value is 0 (disabled) 1 - Enable Local Programming Attribute default value is 1 (enabled)
1	0 –Keypad Interface default access is disabled 1 - Keypad Interface default access is enabled
2	0 - RF Interface default access is disabled 1 - RF Interface default access is enabled
5	0 – Sound Volume attribute default value is 0 (Slight Mode) 1 – Sound Volume attribute default value is equal to something other than 0x00
6	0 – Auto Relock Time attribute default value = 0x00 1 – Auto Relock Time attribute default value is equal to something other than 0x00
7	0 – Led Settings attribute default value = 0x00 1 – Led Settings attribute default value is equal to something other than 0x00

7.3.2.12.9 EnableLocalProgramming Attribute

Enable/disable local programming on the door lock. The local programming features includes but not limited to adding new user codes, deleting existing user codes, add new schedule, deleting existing schedule on the local door lock interfaces. If this value is set to 0x01 or TRUE then local programming is enabled on the door lock. If it is set to 0x00 or FALSE then local programming is disabled on the door lock. Local programming is enabled by default.

7.3.2.12.10 EnableOneTouchLocking Attribute

Enable/disable the ability to lock the door lock with a single touch on the door lock.

7.3.2.12.11 EnableInsideStatusLED Attribute

Enable/disable an inside LED that allows the user to see at a glance if the door is locked.

7.3.2.12.12 EnablePrivacyModeButton Attribute

Enable/disable a button inside the door that is used to put the lock into privacy mode. When the lock is in privacy mode it cannot be manipulated from the outside.

7.3.2.13 Security Settings Attribute Set

Table 7-20. Security Settings Attribute Set

Id	Description	Type	Access	Def	M/O
0x0030	*WrongCodeEntryLimit*	uint8	R*W P	0	O
0x0031	*UserCodeTemporaryDisableTime*	uint8	R*W P	0	O
0x0032	*SendPINOverTheAir*	bool	R*W P	0	O
0x0033	*RequirePINforRFOperation*	bool	R*W P	0	O
0x0034	*SecurityLevel*	enum8	RP	0	O

7.3.2.13.1 WrongCodeEntryLimit Attribute

The number of incorrect codes or RFID presentment attempts a user is allowed to enter before the door will enter a lockout state. The lockout state will be for the duration of *UserCodeTemporaryDisableTime*.

7.3.2.13.2 UserCodeTemporaryDisableTime Attribute

The number of seconds that the lock shuts down following wrong code entry. 1-255 seconds. Device can shut down to lock user out for specified amount of time. (Makes it difficult to try and guess a PIN for the device.)

7.3.2.13.3 SendPINOverTheAir Attribute

Boolean set to True if it is ok for the door lock server to send PINs over the air. This attribute determines the behavior of the server's TX operation. If it is false, then it is not ok for the device to send PIN in any messages over the air.

The PIN field within any door lock cluster message SHALL keep the first octet unchanged and masks the actual code by replacing with 0xFF. For example (PIN "1234"): If the attribute value is True, 0x04 0x31 0x32 0x33 0x34 SHALL be used in the PIN field in any door lock cluster message payload. If the attribute value is False, 0x04 0xFF 0xFF 0xFF 0xFF SHALL be used.

7.3.2.13.4 RequirePINForRFOperation Attribute

Boolean set to True if the door lock server requires that an optional PINs be included in the payload of RF lock operation events like Lock, Unlock and Toggle in order to function.

7.3.2.13.5 SecurityLevel Attribute

Door locks MAY sometimes wish to implement a higher level of security within the application protocol in addition to the default network security. For instance, a door lock MAY wish to use additional APS security for cluster transactions. This protects the door lock against being controlled by any other devices which have access to the network key.

The Security Level attribute allows the door lock manufacturer to indicate what level of security the door lock requires.

There are two levels of security possible within this cluster:

- 0 = Network Security (default)
- 1 = APS Security

This attribute is read only over the network to protect security method defined by each manufacturer.

However, manufacturer can provide method to modify the security setting locally on the device. The security setting modification will not take effect when the device is in a network.

7.3.2.14 Alarm and Event Masks Attribute Set

Table 7-21. Alarm and Event Masks Attribute Set

Id	Description	Type	Access	Default	M/O
0x0040	*AlarmMask*	map16	RWP	0x0000	O
0x0041	*KeypadOperationEventMask*	map16	RWP	0x0000	O
0x0042	*RFOperationEventMask*	map16	RWP	0x0000	O
0x0043	*ManualOperationEventMask*	map16	RWP	0x0000	O
0x0044	*RFIDOperationEventMask*	map16	RWP	0x0000	O
0x0045	*KeypadProgrammingEventMask*	map16	RWP	0x0000	O
0x0046	*RFProgrammingEventMask*	map16	RWP	0x0000	O
0x0047	*RFIDProgrammingEventMask*	map16	RWP	0x0000	O

7.3.2.14.1 *AlarmMask* Attribute

The alarm mask is used to turn on/off alarms for particular functions, as shown in Table 7-22. Alarms for an alarm group are enabled if the associated alarm mask bit is set. Each bit represents a group of alarms. Entire alarm groups can be turned on or off by setting or clearing the associated bit in the alarm mask.

Table 7-22. Alarm Code Table

Alarm Code	Bitmap Number	Alarm Condition
0x00	0	Deadbolt Jammed
0x01	1	Lock Reset to Factory Defaults
0x02	2	Reserved
0x03	3	RF Module Power Cycled

Alarm Code	Bitmap Number	Alarm Condition
0x04	4	Tamper Alarm – wrong code entry limit
0x05	5	Tamper Alarm - front escutcheon removed from main
0x06	6	Forced Door Open under Door Locked Condition

7.3.2.14.2 KeypadOperationEventMask Attribute

Event mask used to turn on and off the transmission of keypad operation events. This mask DOES NOT apply to the storing of events in the report table.

For detail event mask value, please refer to Table 7-30.

7.3.2.14.3 RFOperationEventMask Attribute

Event mask used to turn on and off the transmission of RF operation events. This mask DOES NOT apply to the storing of events in the report table.

For detail event mask value, please refer to Table 7-31.

7.3.2.14.4 ManualOperationEventMask Attribute

Event mask used to turn on and off manual operation events. This mask DOES NOT apply to the storing of events in the report table.

For detail event mask value, please refer to Table 7-32.

7.3.2.14.5 RFIDOperationEventMask Attribute

Event mask used to turn on and off RFID operation events. This mask DOES NOT apply to the storing of events in the report table.

For detail event mask value, please refer to Table 7-33.

7.3.2.14.6 KeypadProgrammingEventMask Attribute

Event mask used to turn on and off keypad programming events. This mask DOES NOT apply to the storing of events in the report table.

For detail event mask value, please refer to Table 7-36.

7.3.2.14.7 RFProgrammingEventMask Attribute

Event mask used to turn on and off RF programming events. This mask DOES NOT apply to the storing of events in the report table.

For detail event mask value, please refer to Table 7-37.

7.3.2.14.8 RFIDProgrammingEventMask Attribute

Event mask used to turn on and off RFID programming events. This mask DOES NOT apply to the storing of events in the report table.

For detail event mask value, please refer to Table 7-38.

7.3.2.15 Server Commands Received

The commands received by the server are listed in Table 7-23.

Table 7-23. Commands Received by the Server Cluster

Command ID	Description	M/O
0x00	Lock Door	M
0x01	Unlock Door	M
0x02	Toggle	O
0x03	Unlock with Timeout	O
0x04	Get Log Record	O
0x05	Set PIN Code	O
0x06	Get PIN Code	O
0x07	Clear PIN Code	O
0x08	Clear All PIN Codes	O
0x09	Set User Status	O
0x0A	Get User Status	O
0x0B	Set Weekday Schedule	O
0x0C	Get Weekday Schedule	O
0x0D	Clear Weekday Schedule	O
0x0E	Set Year Day Schedule	O
0x0F	Get Year Day Schedule	O
0x10	Clear Year Day Schedule	O
0x11	Set Holiday Schedule	O
0x12	Get Holiday Schedule	O
0x13	Clear Holiday Schedule	O
0x14	Set User Type	O

Command ID	Description	M/O
0x15	Get User Type	O
0x16	Set RFID Code	O
0x17	Get RFID Code	O
0x18	Clear RFID Code	O
0x19	Clear All RFID Codes	O

7.3.2.15.1 Lock Door Command

This command causes the lock device to lock the door. As of HA 1.2, this command includes an optional code for the lock. The door lock MAY require a PIN depending on the value of the [Require PIN for RF Operation attribute].

Figure 7-3. Format of the Lock Door Command

Octets	Variable
Data Type	octstr
Field Name	PIN/RFID Code

7.3.2.15.2 Unlock Door Command

This command causes the lock device to unlock the door. As of HA 1.2, this command includes an optional code for the lock. The door lock MAY require a code depending on the value of the [Require PIN for RF Operation attribute].

Note: If the attribute *AutoRelockTime* is supported the lock will close when the auto relock time has expired.

Figure 7-4. Format of the Unlock Door Command

Octets	Variable
Data Type	octstr
Field Name	PIN/RFID Code

7.3.2.15.3 Toggle Command

Request the status of the lock. As of HA 1.2, this command includes an optional code for the lock. The door lock MAY require a code depending on the value of the [Require PIN for RF Operation attribute].

Figure 7-5. Format of the Toggle Command

Octets	Variable
Data Type	octstr

Field Name	PIN/RFID Code

7.3.2.15.4 Unlock with Timeout Command

This command causes the lock device to unlock the door with a timeout parameter. After the time in seconds specified in the timeout field, the lock device will relock itself automatically. This timeout parameter is only temporary for this message transition only and overrides the default relock time as specified in the [Auto Relock Time attribute] attribute. If the door lock device is not capable of or does not want to support temporary Relock Timeout, it SHOULD not support this optional command.

Figure 7-6. Format of the Unlock with Timeout Command

Octets	1	Variable
Data Type	uint16	octstr
Field Name	Timeout in seconds	PIN/RFID Code

7.3.2.15.5 Get Log Record Command

Request a log record. Log number is between 1 – [Number of Log Records Supported attribute]. If log number 0 is requested then the most recent log entry is returned.

Figure 7-7. Format of the Get Log Record Command

Octets	2
Data Type	uint16
Field Name	Log Index

Log record format: The log record format is defined in the description of the Get Log Record Response command.

7.3.2.15.6 User Status and User Type Values

The following User Status and User Type values are used in the payload of multiple commands:

User Status: Used to indicate what the status is for a specific user ID.

Table 7-24. User Status Value

Enum	**Description**
0	Available
1	Occupied / Enabled
3	Occupied / Disabled
0xff	Not Supported

User Type: Used to indicate what the type is for a specific user ID.

Table 7-25. User Type Value

Enum	Description
0	Unrestricted User (default)
1	Year Day Schedule User
2	Week Day Schedule User
3	Master User
4	Non Access User
0xff	Not Supported

Unrestricted: User has access 24/7 provided proper PIN is supplied (e.g., owner). Unrestricted user type is the default user type.

Year Day Schedule User: User has ability to open lock within a specific time period (e.g., guest).

Week Day Schedule User: User has ability to open lock based on specific time period within a reoccurring weekly schedule (e.g., cleaning worker).

Master User: User has ability to both program and operate the door lock. This user can manage the users and user schedules. In all other respects this user matches the unrestricted (default) user. Master user is the only user that can disable the user interface (keypad, RF, etc…).

Non Access User: User is recognized by the lock but does not have the ability to open the lock. This user will only cause the lock to generate the appropriate event notification to any bound devices.

7.3.2.15.7 Set PIN Code Command

Set a PIN into the lock.

Figure 7-8. Format of the Set PIN Code Command

Octets	2	1	1	Variable
Data Type	uint16	uint8	enum8	octstr
Field Name	User ID	User Status	User Type	PIN

User ID is between 0 - [# of PIN Users Supported attribute]. Only the values 1 (Occupied/Enabled) and 3 (Occupied/Disabled) are allowed for User Status.

7.3.2.15.8 Get PIN Code Command

Retrieve a PIN Code. User ID is between 0 - [# of PIN Users Supported attribute].

Figure 7-9. Format of the Get PIN Code Command

Octets	2
Data Type	uint16
Field Name	User ID

7.3.2.15.9 Clear PIN Code Command

Delete a PIN. User ID is between 0 - [# of PIN Users Supported attribute].

Figure 7-10. Format of the Clear PIN Code Command

Octets	2
Data Type	uint16
Field Name	User ID

Note: If you delete a PIN Code and this user didn't have a RFID Code, the user status is set to "0 Available", the user type is set to the default value and all schedules are also set to the default values.

7.3.2.15.10 Clear All PIN Codes Command

Clear out all PINs on the lock.

Note: *On the server, the clear all PIN codes command SHOULD have the same effect as the Clear PIN Code command with respect to the setting of user status, user type and schedules.*

7.3.2.15.11 Set User Status Command

Set the status of a user ID. User Status value of 0x00 is not allowed. In order to clear a user id, the Clear ID Command SHALL be used. For user status value please refer to User Status Value.

Figure 7-11. Format of the Set User Status Command

Octets	2	1
Data Type	uint16	uint8
Field Name	User ID	User Status

7.3.2.15.12 Get User Status Command

Get the status of a user.

Figure 7-12. Format of the Get User Status Command

Octets	2
Data Type	uint16
Field Name	User ID

7.3.2.15.13 Set Week Day Schedule Command

Set a weekly repeating schedule for a specified user.

Figure 7-13. Format of the Set Week Day Schedule Command

Octets	1	2	1	1	1	1	1
Data Type	uint8	uint16	map8	uint8	uint8	uint8	uint8
Field Name	Schedule ID #	User ID	Days Mask	Start Hour	Start Minute	End Hour	End Minute

Schedule ID: number is between 0 – [# of Week Day Schedules Per User attribute].

User ID: is between 0 – [# of Total Users Supported attribute].

Days Mask: bitmask of the effective days in the order XSFTWTMS.

Figure 7-14. Format of Days Mask Bits

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	Sat	Fri	Thur	Wed	Tue	Mon	Sun

Days mask is listed as bitmask for flexibility to set same schedule across multiple days. For the door lock that does not support setting schedule across multiple days within one command, it SHOULD respond with ZCL INVALID_FIELD (0x85) status when received the set schedule command days bitmask field has multiple days selected.

Start Hour: in decimal format represented by 0x00 – 0x17 (00 to 23 hours).

Start Minute: in decimal format represented by 0x00 – 0x3B (00 to 59 mins).

End Hour: in decimal format represented by 0x00 – 0x17 (00 to 23 hours). End Hour SHALL be equal or greater than Start Hour.

End Minute: in decimal format represented by 0x00 – 0x3B (00 to 59 mins).

If End Hour is equal with Start Hour, End Minute SHALL be greater than Start Minute.

When the Server Device receives the command, the Server Device MAY change the user type to the specific schedule user type. Please refer to section 7.3.2.5, Process for Creating a New User with Schedule, at the beginning of this cluster.

7.3.2.15.14 Get Week Day Schedule Command

Retrieve the specific weekly schedule for the specific user.

Figure 7-15. Format of the Get Week Day Schedule Command

Octets	1	2
Data Type	uint8	uint16
Field Name	Schedule ID	User ID

7.3.2.15.15 Clear Week Day Schedule Command

Clear the specific weekly schedule for the specific user.

Figure 7-16. Format of the Clear Week Day Schedule Command

Octets	1	2
Data Type	uint8	uint16
Field Name	Schedule ID	User ID

7.3.2.15.16 Set Year Day Schedule Command

Set a time-specific schedule ID for a specified user.

Figure 7-17. Format of the Set Year Day Schedule Command

Octets	1	2	4	4
Data Type	uint8	uint16	uint32	uint32
Field Name	Schedule ID	User ID	Local Start Time	Local End Time

Schedule ID number is between 0 – [# of Year Day Schedules Supported Per User attribute]. User ID is between 0 – [# of Total Users Supported attribute].

Start time and end time are given in LocalTime. End time must be greater than the start time.

When the Server Device receives the command, the Server Device MAY change the user type to the specific schedule user type. Please refer to Process for Creating a New User with Schedule at the beginning of this cluster.

7.3.2.15.17 Get Year Day Schedule Command

Retrieve the specific year day schedule for the specific user.

Figure 7-18. Format of the Get Year Day Schedule Command

Octets	1	2
Data Type	uint8	uint16
Field Name	Schedule ID	User ID

7.3.2.15.18 Clear Year Day Schedule Command

Clears the specific year day schedule for the specific user.

Figure 7-19. Format of the Clear Year Day Schedule Command

Octets	1	2
Data Type	uint8	uint16
Field Name	Schedule ID	User ID

7.3.2.15.19 Set Holiday Schedule Command

Set the holiday Schedule by specifying local start time and local end time with respect to any Lock Operating Mode.

Figure 7-20. Format of the Set Holiday Schedule Command

Octets	1	4	4	1
Data Type	uint8	uint32	uint32	enum8
Field Name	Holiday Schedule ID	Local Start Time	Local End Time	Operating Mode During Holiday

Holiday Schedule ID number is between 0 – [# of Holiday Schedules Supported attribute].

Start time and end time are given in LocalTime. End time must be greater than the start time.

Operating Mode is valid enumeration value as listed in operating mode attribute.

7.3.2.15.20 Get Holiday Schedule Command

Get the holiday Schedule by specifying Holiday ID.

Figure 7-21. Format of the Get Holiday Schedule Command

Octets	1
Data Type	uint8
Field Name	Holiday Schedule ID

7.3.2.15.21 Clear Holiday Schedule Command

Clear the holiday Schedule by specifying Holiday ID.

Figure 7-22. Format of the Clear Holiday Schedule Command

Octets	1
Data Type	uint8
Field Name	Holiday Schedule ID

7.3.2.15.22 Set User Type Command

Set the type byte for a specified user.

For user type value please refer to User Type Value.

Figure 7-23. Format of the Set User Type Command

Octets	2	1
Data Type	uint16	enum8
Field Name	User ID	User Type

7.3.2.15.23 Get User Type Command

Retrieve the type byte for a specific user.

Figure 7-24. Format of the Get User Type Command

Octets	2
Data Type	uint16
Field Name	User ID

7.3.2.15.24 Set RFID Code Command

Set an ID for RFID access into the lock.

Figure 7-25. Format of the Set RFID Code Command

Octets	2	1	1	Variable
Data Type	uint16	uint8	enum8	octstr
Field Name	User ID	User Status	User Type	RFID Code

User ID: Between 0 **-** [# of RFID Users Supported attribute]. Only the values 1 (Occupied/Enabled) and 3 (Occupied/Disabled) are allowed for User Status.

User Status: Used to indicate what the status is for a specific user ID. The values are according to “Set PIN” while not all are supported.

Table 7-26. User Status Byte Values for Set RFID Code Command

User Status Byte	Value
Occupied / Enabled (Access Given)	1
Occupied / Disabled	3
Not Supported	0xff

User Type: The values are the same as used for “Set PIN Code.”

7.3.2.15.25 Get RFID Code Command

Retrieve an ID. User ID is between 0 **-** [# of RFID Users Supported attribute].

Figure 7-26. Format of the Get RFID Code Command

Octets	2
Data Type	uint16
Field Name	User ID

7.3.2.15.26 Clear RFID Code Command

Delete an ID. User ID is between 0 **-** [# of RFID Users Supported attribute]. If you delete a RFID code and this user didn't have a PIN code, the user status has to be set to "0 Available", the user type has to be set to the default value, and all schedules which are supported have to be set to the default values.

Figure 7-27. Format of the Clear RFID Code Command

Octets	2
Data Type	uint16
Field Name	User ID

7.3.2.15.27 Clear All RFID Codes Command

Clear out all RFIDs on the lock. If you delete all RFID codes and this user didn't have a PIN code, the user status has to be set to "0 Available", the user type has to be set to the default value, and all schedules which are supported have to be set to the default values.

7.3.2.16 Server Commands Generated

The commands generated by the server are listed in Table 7-27.

Table 7-27. Commands Generated by the Server Cluster

Command ID	Description	M/O
0x00	Lock Door Response	M
0x01	Unlock Door Response	M
0x02	Toggle Response	O
0x03	Unlock with Timeout Response	O
0x04	Get Log Record Response	O
0x05	Set PIN Code Response	O
0x06	Get PIN Code Response	O
0x07	Clear PIN Code Response	O
0x08	Clear All PIN Codes Response	O
0x09	Set User Status Response	O
0x0A	Get User Status Response	O
0x0B	Set Weekday Schedule Response	O
0x0C	Get Weekday Schedule Response	O
0x0D	Clear Weekday Schedule Response	O

Command ID	Description	M/O
0x0E	Set Year Day Schedule Response	O
0x0F	Get Year Day Schedule Response	O
0x10	Clear Year Day Schedule Response	O
0x11	Set Holiday Schedule Response	O
0x12	Get Holiday Schedule Response	O
0x13	Clear Holiday Schedule Response	O
0x14	Set User Type Response	O
0x15	Get User Type Response	O
0x16	Set RFID Code Response	O
0x17	Get RFID Code Response	O
0x18	Clear RFID Code Response	O
0x19	Clear All RFID Codes Response	O
0x20	Operating Event Notification	O
0x21	Programming Event Notification	O

7.3.2.16.1 Lock Door Response Command

This command is sent in response to a Lock command with one status byte payload. The Status field SHALL be set to SUCCESS or FAILURE (see 2.5).

The status byte only indicates if the message has received successfully. To determine the lock and/or door status, the client SHOULD query to [Lock State attribute] and [Door State attribute].

Figure 7-28. Format of the Lock Door Response Command Payload

Bits	8
Data Type	enum8
Field Name	Status

7.3.2.16.2 Unlock Door Response Command

This command is sent in response to a Toggle command with one status byte payload. The Status field SHALL be set to SUCCESS or FAILURE (see 2.5).

The status byte only indicates if the message has received successfully. To determine the lock and/or door status, the client SHOULD query to [Lock State attribute] and [Door State attribute].

Figure 7-29. Format of the Unlock Door Response Command Payload

Bits	8
Data Type	enum8
Field Name	Status

7.3.2.16.3 Toggle Response Command

This command is sent in response to a Toggle command with one status byte payload. The Status field SHALL be set to SUCCESS or FAILURE (see 2.5).

The status byte only indicates if the message has received successfully. To determine the lock and/or door status, the client SHOULD query to [Lock State attribute] and [Door State attribute].

7.3.2.16.4 Unlock with Timeout Response Command

This command is sent in response to an Unlock with Timeout command with one status byte payload. The Status field SHALL be set to SUCCESS or FAILURE (see 2.5).

The status byte only indicates if the message has received successfully. To determine the lock and/or door status, the client SHOULD query to [Lock State attribute] and [Door State attribute].

7.3.2.16.5 Get Log Record Response Command

Returns the specified log record. If an invalid log entry ID was requested, it is set to 0 and the most recent log entry will be returned.

Figure 7-30. Format of the Get Log Record Response Command

Octets	2	4	1	1	1	2	Variable
Data Type	uint16	uint32	enum8	uint8	uint8	uint16	octstr[113]
Field Name	Log Entry ID	Timestamp	Event Type	Source (see Operation Event Sources)	Event ID/Alarm Code (see Operation Event Codes)	User ID	PIN

Log Entry ID: the index into the log table where this log entry is stored. If the log entry requested is 0, the most recent log is returned with the appropriate log entry ID.

Timestamp: A LocalTime used to timestamp all events and alarms on the door lock.

Event Type: Indicates the type of event that took place on the door lock.

0x00 = Operation

0x01 = Programming

0x02 = Alarm

Source: A source value where available sources are:

0x00 = Keypad

0x01 = RF

0x02 = Manual

0x03 = RFID

0xff = Indeterminate

If the Event type is 0x02 (Alarm) then the source SHOULD be but does not have to be 0xff (Indeterminate).

Event ID: A one byte value indicating the type of event that took place on the door lock depending on the event code table provided for a given event type and source.

User ID: A two byte value indicating the ID of the user who generated the event on the door lock if one is available. If none is available, 0xffff has to be used.

PIN / ID: A string indicating the PIN code or RFID code that was used to create the event on the door lock if one is available.

7.3.2.16.6 Set PIN Code Response Command

Returns status of the PIN set command. Possible values are:

0 = Success

1 = General failure

2 = Memory full

3 = Duplicate Code error

[113] CCB 2430 PIN/RFID type is Octet String (octstr)

Figure 7-31. Format of the Set PIN Code Response Command

Octets	1
Data Type	uint8
Field Name	Status

7.3.2.16.7 Get PIN Code Response Command

Returns the PIN for the specified user ID.

Figure 7-32. Format of the Get PIN Code Response Command

Octets	2	1	1	Variable
Data Type	uint16	uint8	enum8	octstr
Field Name	User ID	User Status	User Type	Code

If the requested UserId is valid and the Code doesn't exist, Get RFID Code Response SHALL have the following format:

UserId = requested UserId

UserStatus = 0 (available)

UserType = 0xFF (not supported)

RFID = 0 (zero length)

If the requested UserId is invalid, send Default Response with an error status not equal to ZCL_SUCCESS(0x00).

7.3.2.16.8 Clear PIN Code Response Command

Returns pass/fail of the command.

Figure 7-33. Format of the Clear PIN Code Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.9 Clear All PIN Codes Response Command

Returns pass/fail of the command.

Figure 7-34. Format of the Clear All PIN Codes Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.10 Set User Status Response Command

Returns the pass or fail value for the setting of the user status.

Figure 7-35. Format of the Set User Status Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.11 Get User Status Response Command

Returns the user status for the specified user ID.

Figure 7-36. Format of the Get User Status Response Command

Octets	2	1
Data Type	uint16	uint8
Field Name	User ID	User Status

7.3.2.16.12 Set Week Day Schedule Response Command

Returns pass/fail of the command.

Figure 7-37. Format of the Set Week Day Schedule Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.13 Get Week Day Schedule Response Command

Returns the weekly repeating schedule data for the specified schedule ID.

Figure 7-38. Format of the Get Week Day Schedule Response Command

Octets	1	2	1	0/1	0/1	0/1	0/1	0/1
Data Type	uint8	uint16	uint8	uint8	uint8	uint8	uint8	uint8
Field Name	Schedule ID	User ID	Status	Days Mask	Start Hour	Start Minute	End Hour	End Minute

7.3.2.16.13.1 Schedule ID Field

The requested Schedule ID.

7.3.2.16.13.2 User ID Field

The requested User ID.

7.3.2.16.13.3 Status

ZCL SUCCESS (0x00) if both Schedule ID and User ID are valid and there is a corresponding schedule entry.

ZCL INVALID_FIELD (0x85) if either Schedule ID and/or User ID values are not within valid range

ZCL NOT_FOUND (0x8B) if both Schedule ID and User ID are within the valid range, however, there is not corresponding schedule entry found.

Only if the status is ZCL SUCCESS that other remaining fields are included. For other (error) status values, only the fields up to the status field SHALL be present.

7.3.2.16.13.4 Days Mask

Days mask is a bitmask of the effective days in the order [E]SMT WTFS. Bit 7 indicates the enabled status of the schedule ID, with the lower 7 bits indicating the effective days mask.

Figure 7-39. Format of Days Mask Bits

7	**0**	**1**	**2**	**3**	**4**	**5**	**6**
EN	Sun	Mon	Tue	Wed	Thu	Fri	Sat

Bit 7: Enabled status: 1=enabled, 0=disabled

7.3.2.16.13.5 Start Hour

The Start Hour of the Week Day Schedule: 0-23

7.3.2.16.13.6 Start Minute

The Start Min of the Week Day Schedule: 0-59

7.3.2.16.13.7 End Hour

The End Hour of the Week Day Schedule: 0-23, must be greater than Start Hour

7.3.2.16.13.8 End Minute

The End Min of the Week Day Schedule: 0-59

7.3.2.16.14 Clear Week Day Schedule ID Response Command

Returns pass/fail of the command.

Figure 7-40. Format of the Clear Week Day Schedule ID Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.15 Set Year Day Schedule Response Command

Returns pass/fail of the command.

Figure 7-41. Format of the Set Year Day Schedule Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.16 Get Year Day Schedule Response Command

Returns the weekly repeating schedule data for the specified schedule ID.

Figure 7-42. Format of the Get Year Day Schedule Response Command

Octets	1	2	1	0/4	0/4
Data Type	uint8	uint16	uint8	uint32	uint32
Field Name	Schedule ID	User ID	Status	Local Start Time	Local End Time

7.3.2.16.16.1 Schedule ID Field

The requested Schedule ID.

7.3.2.16.16.2 User ID Field

The requested User ID.

7.3.2.16.16.3 Status

ZCL SUCCESS (0x00) if both Schedule ID and User ID are valid and there is a corresponding schedule entry.

ZCL INVALID_FIELD (0x85) if either Schedule ID and/or User ID values are not within valid range

ZCL NOT_FOUND (0x8B) if both Schedule ID and User ID are within the valid range, however, there is not corresponding schedule entry found.

Only if the status is ZCL SUCCESS that other remaining fields are included. For other (error) status values, only the fields up to the status field SHALL be present.

7.3.2.16.16.4 Local Start Time

Start Time of the Year Day Schedule representing by LocalTime.

7.3.2.16.16.5 Local End Time

End Time of the Year Day Schedule representing by LocalTime.

7.3.2.16.17 Clear Year Day Schedule Response Command

Returns pass/fail of the command.

Figure 7-43. Format of the Clear Year Day Schedule Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.18 Set Holiday Schedule Response Command

Returns pass/fail of the command.

Figure 7-44. Format of the Set Holiday Schedule Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.19 Get Holiday Schedule Response Command

Returns the Holiday Schedule Entry for the specified Holiday ID.

Figure 7-45. Format of the Get Holiday Schedule Response Command

Octets	1	1	0/4	0/4	0/1
Data Type	uint8	uint8	uint32	uint32	enum8
Field Name	Holiday Schedule ID	Status	Local Start Time	Local End Time	Operating Mode During Holiday

7.3.2.16.19.1 Holiday Schedule ID

The requested Holiday Schedule ID

7.3.2.16.19.2 Status

ZCL SUCCESS (0x00) if both Schedule ID and User ID are valid and there is a corresponding schedule entry.

ZCL INVALID_FIELD (0x85) if either Schedule ID and/or User ID values are not within valid range

ZCL NOT_FOUND (0x8B) if both Schedule ID and User ID are within the valid range, however, there is not corresponding schedule entry found.

Only if the status is ZCL SUCCESS that other remaining fields are included. For other (error) status values, only the fields up to the status field SHALL be present.

7.3.2.16.19.3 Local Start Time

Start Time of the Year Day Schedule representing by LocalTime.

7.3.2.16.19.4 Local End Time

End Time of the Year Day Schedule representing by LocalTime.

7.3.2.16.19.5 Operating Mode

Operating Mode is valid enumeration value as listed in operating mode attribute.

7.3.2.16.20 Clear Holiday Schedule Response Command

Returns pass/fail of the command.

Figure 7-46. Format of the Clear Holiday Schedule Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.21 Set User Type Response Command

Returns the pass or fail value for the setting of the user type.

Figure 7-47. Format of the Set User Type Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.22 Get User Type Response Command

Returns the user type for the specified user ID.

Figure 7-48. Format of the Get User Type Response Command

Octets	2	1
Data Type	uint16	enum8
Field Name	User ID	User Type

7.3.2.16.23 Set RFID Code Response Command

Returns status of the Set RFID Code command. Possible values are:

0 = Success

1 = General failure

2 = Memory full

3 = Duplicate ID error

Figure 7-49. Format of the Set RFID Code Response Command

Octets	1
Data Type	uint8
Field Name	Status

7.3.2.16.24 Get RFID Code Response Command

Returns the RFID code for the specified user ID.

Figure 7-50. Format of the Get RFID Code Response Command

Octets	2	1	1	Variable
Data Type	uint16	uint8	enum8	octstr
Field Name	User ID	User Status	User Type	RFID Code

If the requested UserId is valid and the Code doesn't exist, Get RFID Code Response SHALL have the following format:

UserId = requested UserId

UserStatus = 0 (available)

UserType = 0xFF (not supported)

RFID = 0 (zero length)

If requested UserId is invalid, send Default Response with an error status not equal to ZCL_SUCCESS(0x00).

7.3.2.16.25 Clear RFID Code Response Command

Returns pass/fail of the command.

Figure 7-51. Format of the Clear RIFD Code Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.26 Clear All RFID Codes Response Command

Returns pass/fail of the command.

Figure 7-52. Format of the Clear All RFID Codes Response Command

Octets	1
Data Type	uint8
Field Name	Status
Field Value	0=pass 1=fail

7.3.2.16.27 Operation Event Notification Command

The door lock server sends out operation event notification when the event is triggered by the various event sources. The specific operation event will only be sent out if the associated bitmask is enabled in the various attributes in the Event Masks Attribute Set.

All events are optional.

Figure 7-53. Format of the Operation Event Notification Command

Octets	1	1	2	1	4	Variable/0
Data Type	uint8	uint8	uint16	octstr[114]	uint32	string
Field Name	Operation Event Source	Operation Event Code	User ID	PIN	LocalTime	Data

7.3.2.16.27.1 Operation Event Sources

This field indicates where the event was triggered from.

Table 7-28. Operation Event Source Value

Value	Source
0x00	Keypad
0x01	RF
0x02	Manual
0x03	RFID
0xFF	Indeterminate

7.3.2.16.27.2 Operation Event Codes

[114] CCB 2430 PIN/RFID type is Octet String (octstr)

The door lock optionally sends out notifications (if they are enabled) whenever there is a significant operational event on the lock. When combined with a source from the Event Source table above, the following operational event codes constitute an event on the door lock that can be both logged and sent to a bound device using the Operation Event Notification command.

Not all operation event codes are applicable to each of the event source. Table 7-29 marks each event code with "A" if the event code is applicable to the event source.

Table 7-29. Operation Event Code Value

Value	Operation Event Code	Keypad	RF	Manual	RFID
0x00	UnknownOrMfgSpecific	A	A	A	A
0x01	Lock	A	A	A	A
0x02	Unlock	A	A	A	A
0x03	LockFailureInvalidPINorID	A	A		A
0x04	LockFailureInvalidSchedule	A	A		A
0x05	UnlockFailureInvalidPINorID	A	A		A
0x06	UnlockFailureInvalidSchedule	A	A		A
0x07	OneTouchLock			A	
0x08	KeyLock			A	
0x09	KeyUnlock			A	
0x0A	AutoLock			A	
0x0B	ScheduleLock			A	
0x0C	ScheduleUnlock			A	
0x0D	Manual Lock (Key or Thumbturn)			A	
0x0E	Manual Unlock (Key or Thumbturn)			A	
0x0F	Non-Access User Operational Event	A			

7.3.2.16.27.3 User ID

The User ID who performed the event.

7.3.2.16.27.4 PIN

The PIN that is associated with the User ID who performed the event.

7.3.2.16.27.5 LocalTime

The LocalTime that indicates when the event is triggered. If time is not supported, the field SHALL be populated with default not used value 0xFFFFFFFF.

7.3.2.16.27.6 Data

The operation event notification command contains a variable string, which can be used to pass data associated with a particular event. Generally this field will be left empty. However, manufacturer can choose to use this field to store/display manufacturer-specific information.

7.3.2.16.27.7 Keypad Operation Event Notification

Keypad Operation Event Notification feature is enabled by setting the associated bitmasks in the [Keypad Operation Event Mask attribute].

Table 7-30. Keypad Operation Event Value

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x00	0x00	BIT(0)	Unknown or manufacturer-specific keypad operation event
0x00	0x01	BIT(1)	Lock, source: keypad
0x00	0x02	BIT(2)	Unlock, source: keypad
0x00	0x03	BIT(3)	Lock, source: keypad, error: invalid PIN
0x00	0x04	BIT(4)	Lock, source: keypad, error: invalid schedule
0x00	0x05	BIT(5)	Unlock, source: keypad, error: invalid code
0x00	0x06	BIT(6)	Unlock, source: keypad, error: invalid schedule
0x00	0x0F	BIT(7)	Non-Access User operation event, source keypad.

7.3.2.16.27.8 RF Operation Event Notification

RF Operation Event Notification feature is enabled by setting the associated bitmasks in the [RF Operation Event Mask attribute].

Table 7-31. RF Operation Event Value

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x01	0x00	BIT(0)	Unknown or manufacturer-specific RF operation event
0x01	0x01	BIT(1)	Lock, source: RF
0x01	0x02	BIT(2)	Unlock, source: RF
0x01	0x03	BIT(3)	Lock, source: RF, error: invalid code
0x01	0x04	BIT(4)	Lock, source: RF, error: invalid schedule
0x01	0x05	BIT(5)	Unlock, source: RF, error: invalid code

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x01	0x06	BIT(6)	Unlock, source: RF, error: invalid schedule

7.3.2.16.27.9 Manual Operation Event Notification

Manual Operation Event Notification feature is enabled by setting the associated bitmasks in the [Manual Operation Event Mask attribute] attribute.

Table 7-32. Manual Operation Event Value

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x02	0x00	BIT(0)	Unknown or manufacturer-specific manual operation event
0x02	0x01	BIT(1)	Thumbturn Lock
0x02	0x02	BIT(2)	Thumbturn Unlock
0x02	0x07	BIT(3)	One touch lock
0x02	0x08	BIT(4)	Key Lock
0x02	0x09	BIT(5)	Key Unlock
0x02	0x0A	BIT(6)	Auto lock
0x02	0x0B	BIT(7)	Schedule Lock
0x02	0x0C	BIT(8)	Schedule Unlock
0x02	0x0D	BIT(9)	Manual Lock (Key or Thumbturn)
0x02	0x0E	BIT(10)	Manual Unlock (Key or Thumbturn)

7.3.2.16.27.10 RFID Operation Event Notification

RFID Operation Event Notification feature is enabled by setting the associated bitmasks in the [RFID Operation Event Mask attribute].

Table 7-33. RFID Operation Event Value

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x03	0x00	BIT(0)	Unknown or manufacturer-specific keypad operation event
0x03	0x01	BIT(1)	Lock, source: RFID
0x03	0x02	BIT(2)	Unlock, source: RFID

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x03	0x03	BIT(3)	Lock, source: RFID, error: invalid RFID ID
0x03	0x04	BIT(4)	Lock, source: RFID, error: invalid schedule
0x03	0x05	BIT(5)	Unlock, source: RFID, error: invalid RFID ID
0x03	0x06	BIT(6)	Unlock, source: RFID, error: invalid schedule

7.3.2.16.28 Programming Event Notification Command

The door lock server sends out a programming event notification whenever a programming event takes place on the door lock.

As with operational events, all programming events can be turned on and off by flipping bits in the associated event mask.

The programming event notification command includes an optional string of data that can be used by the manufacturer to pass some manufacturer-specific information if that is required.

Figure 7-54. Format of the Programming Event Notification Command

Octets	1	1	2	1	1	1	4	Variable/0
Data Type	uint8	uint8	uint16	octstr[115]	enum8	uint8	uint32	string
Field Name	Program Event Source	Program Event Code	User ID	PIN	User Type	User Status	LocalTime	Data

7.3.2.16.28.1 Operation Event Sources

This field indicates where the event was triggered from.

Table 7-34. Operation Event Source Value

Value	Source
0x00	Keypad
0x01	RF
0x02	Reserved (Manual in Operation Event)
0x03	RFID
0xFF	Indeterminate

7.3.2.16.28.2 Programming Event Codes

[115] CCB 2430 PIN/RFID type is Octet String (octstr)

The door lock optionally sends out notifications (if they are enabled) whenever there is a significant programming event on the lock. When combined with a source from the Event Source table above, the following programming event codes constitute an event on the door lock that can be both logged and sent to a bound device using the Programming Event Notification command.

Not all event codes are applicable to each of the event source. Table 7-35 marks each event code with “A” if the event code is applicable to the event source.

Table 7-35. Programming Event Codes

Value	Programming Event Code	Keypad	RF	RFID
0x00	UnknownOrMfgSpecific	A	A	A
0x01	MasterCodeChanged	A		
0x02	PINCodeAdded	A	A	
0x03	PINCodeDeleted	A	A	
0x04	PINCodeChanged	A	A	
0x05	RFIDCodeAdded			A
0x06	RFIDCodeDeleted			A

7.3.2.16.28.3 User ID

The User ID who performed the event

7.3.2.16.28.4 PIN

The PIN that is associated with the User ID who performed the event

7.3.2.16.28.5 User Type

The User Type that is associated with the User ID who performed the event

7.3.2.16.28.6 User Status

The User Status that is associated with the User ID who performed the event

7.3.2.16.28.7 LocalTime

The LocalTime that indicates when the event is triggered. If time is not supported, the field SHALL be populated with default not used value 0xFFFFFFFF.

7.3.2.16.28.8 Data

The programming event notification command contains a variable string, which can be used to pass data associated with a particular event. Generally this field will be left empty. However, manufacturer can choose to use this field to store/display manufacturer-specific information.

7.3.2.16.28.9 Keypad Programming Event Notification

Keypad Programming Event Notification feature is enabled by setting the associated bitmasks in the [Keypad Programming Event Mask attribute].

Table 7-36. Keypad Programming Event Value

Event Source	Program Event Code	Attribute Bitmask	Event Description
0x00	0x00	BIT(0)	Unknown or manufacturer-specific keypad programming event
0x00	0x01	BIT(1)	Master code changed, source: keypad User ID: master user ID. PIN: default or master code if codes can be sent over the air per attribute. User type: default User Status: default
0x00	0x02	BIT(2)	PIN added, source: keypad User ID: user ID that was added. PIN: code that was added (if codes can be sent over the air per attribute.) User type: default or type added. User Status: default or status added.
0x00	0x03	BIT(3)	PIN deleted, source: keypad User ID: user ID that was deleted. PIN: code that was deleted (if codes can be sent over the air per attribute.) User type: default or type deleted. User Status: default or status deleted.
0x00	0x04	BIT(4)	PIN changed Source: keypad User ID: user ID that was changed PIN: code that was changed (if codes can be sent over the air per attribute.) User type: default or type changed. User Status: default or status changed.

7.3.2.16.28.10 RF Programming Event Notification

RF Programming Event Notification feature is enabled by setting the associated bitmasks in the [RF Programming Event Mask attribute].

Table 7-37. RF Programming Event Value

Event Source	Program Event Code	Attribute Bitmask	Event Description
0x01	0x00	BIT(0)	Unknown or manufacturer-specific RF programming event.
0x01	0x02	BIT(2)	PIN added, source RF Same as keypad source above

Event Source	Program Event Code	Attribute Bitmask	Event Description
0x01	0x03	BIT(3)	PIN deleted, source RF Same as keypad source above.
0x01	0x04	BIT(4)	PIN changed Source RF Same as keypad source above
0x01	0x05	BIT(5)	RFID code added, Source RF
0x01	0x06	BIT(6)	RFID code deleted, Source RF

7.3.2.16.28.11 RFID Programming Event Notification

RFID Programming Event Notification feature is enabled by setting the associated bitmasks in the [RFID Programming Event Mask attribute].

Table 7-38. RFID Programming Event Value

Event Source	Program Event Code	Attribute Bitmask	Event Description
0x03	0x00	BIT(0)	Unknown or manufacturer-specific keypad programming event
0x03	0x05	BIT(5)	ID Added, Source: RFID User ID: user ID that was added. ID: ID that was added (if codes can be sent over the air per attribute.) User Type: default or type added. User Status: default or status added.
0x03	0x06	BIT(6)	ID Deleted, Source: RFID User ID: user ID that was deleted. ID: ID that was deleted (if codes can be sent over the air per attribute.) User Type: default or type deleted. User Status: default or status deleted.

7.3.2.17 Scene Table Extension

If the Scene server cluster is implemented, the following extension field is added to the Scene table:

- **LockState**

  When the *LockState* attribute is part of a Scene table, the attribute is treated as a writable command; that is, setting the *LockState* to lock will command the lock to lock, and setting the *LockState* to unlock will command the lock to unlock. Setting the *LockState* attribute to "not fully locked" is not supported.

  The Transition Time field in the Scene table will be treated as a delay before setting the *LockState* attribute; that is, it is possible to activate a scene with the lock actuation some seconds later.

Locks that do not have an actuation mechanism SHOULD not support the Scene table extension.

7.3.3 Client

The client supports no cluster specific attributes. The client receives the cluster-specific commands generated by the server, as shown in Table 7-27. The client generates the cluster-specific commands that will be received by the server, as shown in Table 7-23.

7.4 Window Covering

7.4.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The window covering cluster provides an interface for controlling and adjusting automatic window coverings such as drapery motors, automatic shades, and blinds.

7.4.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added; CCB 1994 1995 1996 1997 2086 2094 2095 2096 2097
2	CCB 2328

7.4.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	WNCV	Type 1 (client to server)

7.4.1.3 Cluster Identifiers

Identifier	Name
0x0102	Window Covering

7.4.2 Server

7.4.2.1 Attributes

For convenience, the attributes defined in this cluster are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 7-39.

Table 7-39. Window Covering Attribute Set

Attribute Set Identifier	Description
0x00	Window Covering Information
0x01	Window Covering Settings

7.4.2.1.1 Window Covering Information Attribute Set

The Window Covering Information attribute set contains the attributes summarized in Table 7-40.

Table 7-40. Window Covering Information Attribute Set

Id	Name	Unit	Type	Range	Acc	Default	M/O
0x0000	*WindowCoveringType*		enum8	0x00 – 0x09	R	0x00	M
0x0001	*PhysicalClosedLimit – Lift*	*cm*	uint16	0x0000 – 0xffff	R	0x0000	O
0x0002	*PhysicalClosedLimit – Tilt*	*0.1°*	uint16	0x0000 – 0xffff	R	0x0000	O
0x0003	*CurrentPosition – Lift*	*cm*	uint16	0x0000 – 0xffff	R	0x0000	O
0x0004	*Current Position – Tilt*	*0.1°*	uint16	0x0000 – 0xffff	R	0x0000	O
0x0005	*Number of Actuations – Lift*		uint16	0x0000 – 0xffff	R	0x0000	O
0x0006	*Number of Actuations – Tilt*		uint16	0x0000 – 0xffff	R	0x0000	O
0x0007	*Config/Status*		map8	0xxx xxxx	R	0000 0011	M
0x0008	*Current Position Lift Percentage*		uint8	0-0x64	RSP	0x00	**M***
0x0009	*Current Position Tilt Percentage*		uint8	0-0x64	RSP	0x00	**M***

***Note:** *"M*" designates that the related attributes are required to be mandatory only if Closed Loop control is enabled and Lift/Tilt actions are correspondingly supported, i.e. the CurrentPositionLiftPercentage attribute SHALL be mandatory only if Closed Loop control and Lift actions are supported; and the CurrentPositionTiltPercentage attribute SHALL be mandatory only if Closed Loop control and Tilt actions are supported.*

7.4.2.1.2 WindowCoveringType Attribute

The *WindowCoveringType* attribute identifies the type of window covering being controlled by this endpoint and SHALL be set to one of the non-reserved values in Table 7-41.

Table 7-41. Window Covering Type

Value	Window Covering Type	Supported Actions
0x00	Rollershade	Lift
0x01	Rollershade - 2 Motor	Lift
0x02	Rollershade – Exterior	Lift
0x03	Rollershade - Exterior - 2 Motor	Lift
0x04	Drapery	Lift
0x05	Awning	Lift
0x06	Shutter	Tilt
0x07	Tilt Blind - Tilt Only	Tilt
0x08	Tilt Blind - Lift and Tilt	Lift, Tilt
0x09	Projector Screen	Lift

7.4.2.1.2.1 PhysicalClosedLimit - Lift Attribute

The *PhysicalClosedLimitLift* attribute identifies the maximum possible encoder position possible (in centimeters) to position the height of the window covering – this is ignored if the device is running in Open Loop Control.

7.4.2.1.2.2 PhysicalClosedLimit - Tilt Attribute

The *PhysicalClosedLimitTilt* attribute identifies the maximum possible encoder position possible (tenth of a degrees) to position the angle of the window covering – this is ignored if the device is running in Open Loop Control.

7.4.2.1.2.3 CurrentPosition - Lift Attribute

The *CurrentPositionLift* attribute identifies the actual position (in centimeters) of the window covering from the top of the shade if Closed Loop Control is enabled. This attribute is ignored if the device is running in Open Loop Control.

7.4.2.1.2.4 Current Position - Tilt Attribute

The *CurrentPositionTilt* attribute identifies the actual tilt position (in tenth of an degree) of the window covering from Open if Closed Loop Control is enabled. This attribute is ignored if the device is running in Open Loop Control.

7.4.2.1.2.5 Number of Actuations - Lift Attribute

The *NumberOfActuationsLift* attribute identifies the total number of lift actuations applied to the Window Covering since the device was installed.

7.4.2.1.2.6 Number of Actuations - Tilt Attribute

The *NumberOfActuationsTilt* attribute identifies the total number of tilt actuations applied to the Window Covering since the device was installed.

7.4.2.1.2.7 Config/Status Attribute

The *ConfigStatus* attribute makes configuration and status information available. To change settings, devices SHALL write to the Mode attribute of the Window Covering Settings Attribute Set. The behavior causing the setting or clearing of each bit is vendor specific. See Table 7-42 for details on each bit.

Table 7-42. Bit Meanings for the Config/Status Attribute

Bit	Meaning	Description
bit0	0 = Not Operational 1 = Operational	Operational: This status bit defines if the Window Covering is operational.
bit1	0 = Not Online 1 = Online	Online: This status bit defines if the Window Covering is enabled for transmitting over the network.
bit2	0 = Commands are normal 1 = Open/Up Commands have been reversed	Reversal – Lift commands: This status bit identifies if the direction of rotation for the Window Covering has been reversed in order for Open/Up commands to match the physical installation condition.
bit3	0 = Lift control is Open Loop 1 = Lift control is Closed Loop	Control – Lift: This status bit identifies if the window covering supports Open Loop or Closed Loop Lift Control
bit4	0 = Tilt control is Open Loop 1 = Tilt control is Closed Loop	Control – Tilt: This status bit identifies if the window covering supports Open Loop or Closed Loop Tilt Control
bit5	0 = Timer Controlled 1 = Encoder Controlled This bit is Ignored if running Lift in Open Loop Control.	Encoder – Lift: This status bit identifies if a Closed Loop Controlled Window Covering is employing an encoder for positioning the height of the window covering.
bit6	0 = Timer Controlled 1 = Encoder Controlled This bit is Ignored if running Tilt in Open Loop Control.	Encoder – Tilt: This status bit identifies if a Closed Loop Controlled Window Covering is employing an encoder for tilting the window covering.

7.4.2.1.2.8 Current Position Lift Percentage Attribute

The *CurrentPositionLiftPercentage* attribute identifies the actual position as a percentage between the *InstalledOpenLimitLift* attribute and the *InstalledClosedLimitLift* attribute of the window covering from the up/open position if Closed Loop Control is enabled. If the device is running in Open Loop Control or the device only supports Tilt actions, this attribute is not required as an attribute but has a special interpretation when received as part of a scene command (see “Scene Table Extensions” below).

7.4.2.1.2.9 Current Position Tilt Percentage Attribute

The *CurrentPositionTiltPercentage* attribute identifies the actual position as a percentage between the *InstalledOpenLimitTilt* attribute and the *InstalledClosedLimitTilt* attribute of the window covering from the up/open position if Closed Loop Control is enabled. If the device is running in Open Loop Control or the device only support Lift actions, this attribute is not required as an attribute but has a special interpretation when received as part of a scene command (see “Scene Table Extensions” below).

7.4.2.1.3 Window Covering Settings Attribute Set

The *WindowCoveringSettings* attribute set contains the attributes summarized in Table 7-43.

Table 7-43. Window Covering Settings Attribute Set

Id	Name	Unit	Type	Range	Acc	Default	M/O
0x0000	*InstalledOpenLimit – Lift*	*cm*	uint16	0x0000 – 0xffff	R	0x0000	M*
0x0001	*InstalledClosedLimit – Lift*	*cm*	uint16	0x0000 – 0xffff	R	0xffff	M*
0x0002	*InstalledOpenLimit – Tilt*	*0.1°*	uint16	0x0000 – 0xffff	R	0x0000	M*
0x0003	*InstalledClosedLimit – Tilt*	*0.1°*	uint16	0x0000 – 0xffff	R	0xffff	M*
0x0004	*Velocity – Lift*	*cm/sec*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0005	*Acceleration Time – Lift*	*0.1sec*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0006	*Deceleration Time – Lift*	*0.1sec*	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0007	*Mode*		map8		RW	0000[116] 0100	M
0x0008	*Intermediate Setpoints – Lift*		octstr	-	RW	“1,0x0000”	O
0x0009	*Intermediate Setpoints – Tilt*		octstr	-	RW	“1,0x0000”	O

Note: *"M*" designates that the related attributes are required to be mandatory only if Closed Loop control is enabled and Lift/Tilt actions are correspondingly supported, i.e. the InstalledOpenLimitLift and InstalledClosedLimitLift attributes SHALL be mandatory only if Closed Loop control and Lift actions are supported; and the InstalledOpenLimitTilt and InstalledClosedLimitTilt attributes SHALL be mandatory only if Closed Loop control and Tilt actions are supported.*

7.4.2.1.3.1 InstalledOpenLimit – Lift

The *InstalledOpenLimitLift* attribute identifies the Open Limit for Lifting the Window Covering whether position (in centimeters) is encoded or timed. This attribute is ignored if the device is running in Open Loop Control or only supports Tilt actions.

7.4.2.1.3.2 InstalledClosedLimit – Lift

The *InstalledClosedLimitLift* attribute identifies the Closed Limit for Lifting the Window Covering whether position (in centimeters) is encoded or timed. This attribute is ignored if the device is running in Open Loop Control or only supports Tilt actions.

7.4.2.1.3.3 InstalledOpenLimit – Tilt

[116] CCB 2328 no range for map and fix default

The *InstalledOpenLimitTilt* attribute identifies the Open Limit for Tilting the Window Covering whether position (in tenth of a degree) is encoded or timed. This attribute is ignored if the device is running in Open Loop Control or only supports Lift actions.

7.4.2.1.3.4 InstalledClosedLimit – Tilt

The *InstalledClosedLimitTilt* attribute identifies the Closed Limit for Tilting the Window Covering whether position (in tenth of a degree) is encoded or timed. This attribute is ignored if the device is running in Open Loop Control or only supports Lift actions.

7.4.2.1.3.5 Velocity – Lift

The *VelocityLift* attribute identifies the velocity (in centimeters per second) associated with Lifting the Window Covering.

7.4.2.1.3.6 Acceleration Time – Lift

The *AccelerationTimeLift* attribute identifies any ramp up times to reaching the velocity setting (in tenth of a second) for positioning the Window Covering.

7.4.2.1.3.7 Deceleration Time – Lift

The *DecelerationTimeLift* attribute identifies any ramp down times associated with stopping the positioning (in tenth of a second) of the Window Covering.

7.4.2.1.3.8 Mode

The *Mode* attribute allows configuration of the Window Covering, such as: reversing the motor direction, placing the Window Covering into calibration mode, placing the motor into maintenance mode, disabling the network, and disabling status LEDs. See Table 7-44 for details.

Table 7-44. Bit Meanings for the Mode Attribute

Bit	Meaning	Description
bit0	0 = motor direction is normal 1 = motor direction is reversed	Disables (0) or Enables (1) the reversal of the motor rotating direction associated with an UP/OPEN command. Should be set so that an UP/OPEN command matches moving the Window Covering physically in that direction.
bit1	0 = run in normal mode 1 = run in calibration mode	Disables (0) or Enables (1) placing the Window Covering into Calibration Mode where limits are either setup using physical tools or limits are learned by the controller based on physical setup of the Window Covering by an installer.
bit2	0 = motor is running normally 1 = motor is running in maintenance mode	Disables (0) or Enables (1) placing the motor into Maintenance Mode where the motor cannot be moved over the network or by a switch connected to a Local Switch Input.
bit3	0 = LEDs are off 1 = LEDs will display feedback	Disables (0) or Enables (1) the display of any feedback LEDs resident especially on the packaging of an endpoint where they may cause distraction to the occupant.

7.4.2.1.3.9 Intermediate Setpoints – Lift

Identifies the number of Intermediate Setpoints supported by the Window Covering for Lift and then identifies the position settings for those Intermediate Setpoints if Closed Loop Control is supported. This is a comma delimited ASCII character string. For example: “2,0x0013, 0x0030”

7.4.2.1.3.10 Intermediate Setpoints – Tilt

Identifies the number of Intermediate Setpoints supported by the Window Covering for Tilt and then identifies the position settings for those Intermediate Setpoints if Closed Loop Control is supported. This is a comma delimited ASCII character string. For example: “2,0x0013, 0x0030”

7.4.2.2 Commands Received

Table 7-45. Commands Received by the Window Covering Server Cluster

Command ID	Description	M/O
0x00	Up / Open	M
0x01	Down / Close	M
0x02	Stop	M
0x04	Go To Lift Value	O
0x05	Go to Lift Percentage	O
0x07	Go to Tilt Value	O
0x08	Go to Tilt Percentage	O

7.4.2.2.1 Up / Open Command

7.4.2.2.1.1 Payload Format

This command has no payload.

7.4.2.2.1.2 Effect on Receipt

Upon receipt of this command, the Window Covering will adjust the window so the physical lift is at the *InstalledOpenLimit – Lift* and the tilt is at the *InstalledOpenLimit – Tilt*. This will happen as fast as possible.

7.4.2.2.2 Down / Close Command

7.4.2.2.2.1 Payload Format

This command has no payload.

7.4.2.2.2.2 Effect on Receipt

Upon receipt of this command, the Window Covering will adjust the window so the physical lift is at the *InstalledClosedLimit – Lift* and the tilt is at the *InstalledClosedLimit – Tilt*. This will happen as fast as possible.

7.4.2.2.3 Stop Command

7.4.2.2.3.1 Payload Format

This command has no payload.

7.4.2.2.3.2 Effect on Receipt

Upon receipt of this command, the Window Covering will stop any adjusting to the physical tilt and lift that is currently occurring.

7.4.2.2.4 Go To Lift Value

7.4.2.2.4.1 Payload Format

The Go To Lift Value command payload SHALL be formatted as illustrated in Figure 7-55.

Figure 7-55. Format of the Go To Lift Value Command

Octets	2
Data Type	uint16
Field Name	Lift Value

7.4.2.2.4.2 Effect on Receipt

Upon receipt of this command, the Window Covering will adjust the window so the physical lift is at the lift value specified in the payload of this command as long as that value is not larger than *InstalledOpenLimit – Lift* and not smaller than *InstalledClosedLimit – Lift*. If the lift value is out of bounds a default response containing the status of INVALID_VALUE will be returned.

7.4.2.2.5 Go to Lift Percentage

7.4.2.2.5.1 Payload Format

The Go To Lift Percentage command payload SHALL be formatted as illustrated in Figure 7-56.

Figure 7-56. Format of the Go To Lift Percentage Command

Octets	1
Data Type	uint8
Field Name	Percentage Lift Value

7.4.2.2.5.2 Effect on Receipt

Upon receipt of this command, the Window Covering will adjust the window so the physical lift is at the lift percentage specified in the payload of this command.The percentage value will be mapped to a 8-bit unsigned integer value between InstalledOpenLimit and InstalledClosedLimit. If the percentage lift value is larger than 100, no physical action will be taken and a default response containing the status of INVALID_VALUE will be returned. If the device only supports open loop lift action then a zero percentage SHOULD be treated as a down/close command and a non-zero percentage SHOULD be treated as an up/open command. If the device is only a tilt control device, then the command SHOULD be ignored and a UNSUPPORTED_COMMAND status SHOULD be returned. The device must support either the Go To Lift Percentage or the Go To Tilt Percentage command.

7.4.2.2.6 Go to Tilt Value

7.4.2.2.6.1 Payload Format

The Go To Tilt Value command payload SHALL be formatted as illustrated in Figure 7-57.

Figure 7-57. Format of the Go To Tilt Value Command

Octets	2
Data Type	uint16
Field Name	Tilt Value

7.4.2.2.6.2 Effect on Receipt

Upon receipt of this command, the Window Covering will adjust the window so the physical tilt is at the tilt value specified in the payload of this command as long as that value is not larger than *InstalledOpenLimit – Tilt* and not smaller than *InstalledClosedLimit – Tilt*. If the tilt value is out of bounds a default response containing the status of INVALID_VALUE will be returned.

7.4.2.2.7 Go to Tilt Percentage

7.4.2.2.7.1 Payload Format

The Go To Tilt Percentage command payload SHALL be formatted as illustrated below.

Figure 7-58. Format of the Go To Lift Percentage Command

Octets	1
Data Type	uint8
Field Name	Percentage Tilt Value

7.4.2.2.7.2 Effect on Receipt

Upon receipt of this command, the Window Covering will adjust the window so the physical tilt is at the tilt percentage specified in the payload of this command. The percentage value will be mapped to a 8-bit unsigned integer value between *InstalledOpenLimit-Tilt* and *InstalledClosedLimit-Tilt*. If the percentage tilt value is larger than 100, no physical action will be taken and a default response containing the status of INVALID_VALUE will be returned. If the device only supports open loop tilt action then a zero percentage SHOULD be treated as a down/close command and a non-zero percentage SHOULD be treated as an up/open command. If the device is only a lift control device, then the command SHOULD be ignored and a UNSUPPORTED_COMMAND status SHOULD be returned. The device must support either the Go To Lift Percentage or the Go To Tilt Percentage command.

7.4.2.3 Commands Generated

This cluster uses the standard Default Response command defined in the ZCL specification for responding to received commands. Possible status values that can be returned are: SUCCESS, NOT_FOUND, NOT_AUTHORIZED, INSUFFICIENT_SPACE, UNSUP_CLUSTER_COMMAND, INVALID_FIELD, INVALID_VALUE, HARDWARE_FAILURE, FAILURE.

7.4.2.4 Scene Table Extensions

If the Window Covering server cluster is implemented, the following extension field is added to the Scene table:

- **CurrentPositionLiftPercentage**
  When the *CurrentPositionLiftPercentage* attribute is part of a Scene table, the attribute is treated as a writeable command, that is, setting the lift percentage of the covering device to the value specified in the scene table extension over the specified transition time. The device MAY treat the command as a linear transition if appropriate or MAY accelerate and decelerate as it deems necessary. If the device is only a tilt controlling device this scene table extension is ignored. If the device is an open loop controlled lift device, then a percentage of 0 is treated as a close command and a non zero percentage is treated as an open command and the device will ignore the transition time and transition as fast as appropriate for that device.

- **CurrentPositionTiltPercentage**
  When the *CurrentPositionTiltPercentage* attribute is part of a Scene table, the attribute is treated as a writeable command, that is, setting the tilt percentage of the covering device to the value specified in the scene table extension over the specified transition time. The device MAY treat the command as a linear transition if appropriate or MAY accelerate and decelerate as it deems necessary. If the device is only a lift controlling device this scene table extension is ignored. If the device is an open loop controlled tilt device, then a percentage of 0 is treated as a close command and a non zero percentage is treated as an open command and the device will ignore the transition time and transition as fast as appropriate for that device.

7.4.2.5 Attribute Reporting

This cluster SHALL support attribute reporting using the Report Attributes command and according to the minimum and maximum reporting interval settings described in the ZCL. The following attributes SHALL be reported:

Current Position - *Lift Percentage*

Current Position - *Tilt Percentage*

7.4.3 Client

The client has no cluster specific attributes. No cluster specific commands are received by the client. The client generates the cluster specific commands detailed in sub-clause 7.4.2.2.

CHAPTER 8 SECURITY AND SAFETY

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

8.1 General Description

8.1.1 Introduction

The clusters specified in this document are for use in security and safety related applications.

The clusters currently defined are those that are used by wireless Intruder Alarm Systems (IAS). Intruder Alarm systems include functions for the detection of intruders and/or triggering, processing of information, notification of alarms and the means to operate the IAS.

Functions additional to those MAY be included in IAS providing they do not influence the correct operation of the mandatory functions. Components of other applications MAY be combined or integrated with a IAS, providing the performance of the IAS components is not adversely influenced.

8.1.2 Cluster List

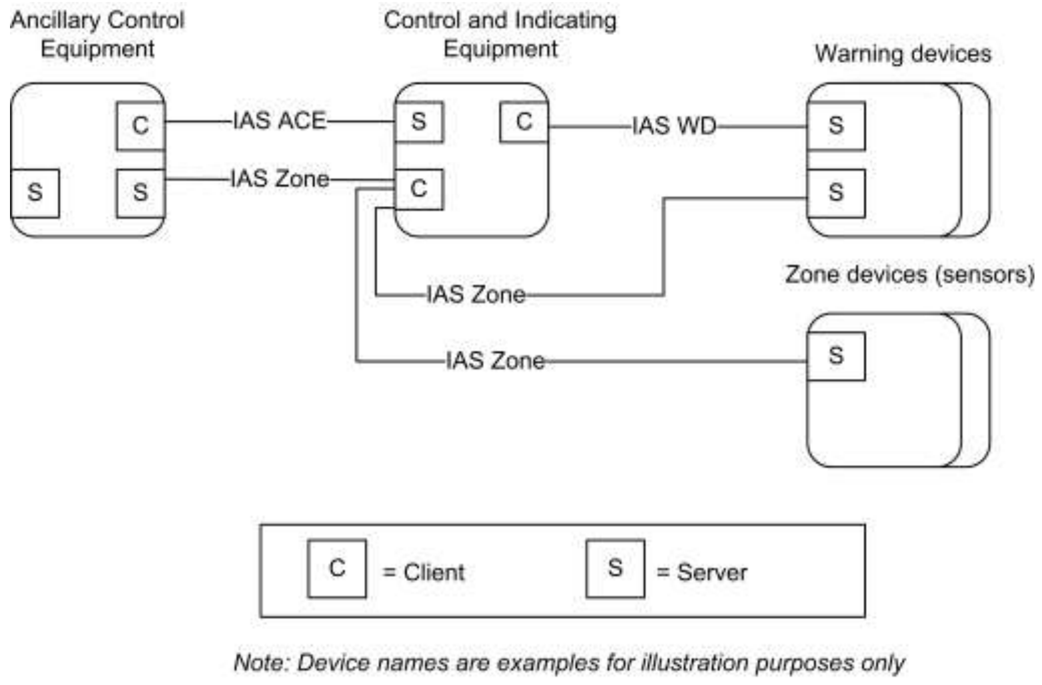
This section lists the clusters specified in this document, and gives examples of typical usage for the purpose of clarification.

The clusters defined in this document are listed in Table 8-1.

Table 8-1. Clusters of the Security and Safety Functional Domain

Cluster ID	Cluster Name	Description
0x500	IAS Zone	Attributes and commands for IAS security zone devices.
0x501	IAS ACE	Attributes and commands for IAS Ancillary Control Equipment.
0x502	IAS WD	Attributes and commands for IAS Warning Devices

Figure 8-1. Typical Usage of the IAS Clusters


Ancillary Control Equipment
Control and Indicating Equipment
Warning devices
C
S
S
IAS ACE
IAS Zone
S
C
C
IAS WD
S
S
Zone devices (sensors)
IAS Zone
IAS Zone
S
C = Client
S = Server
Note: Device names are examples for illustration purposes only

8.2 IAS Zone

8.2.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The IAS Zone cluster defines an interface to the functionality of an IAS security zone device. IAS Zone supports up to two alarm types per zone, low battery reports and supervision of the IAS network.

8.2.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added; ZHA 1.2 and 1.2.1 features; CCB 2044 2045
2	CCB 2352

8.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	IASZ	Type 2 (server to client)

8.2.1.3 Cluster Identifiers

Identifier	Name
0x0500	IAS Zone

8.2.2 Server

8.2.2.1 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 8-2.

Table 8-2. Attribute Sets for the IAS Zone Cluster

Attribute Set Identifier	Description
0x000	Zone information
0x001	Zone settings

8.2.2.1.1 Zone Information Attribute Set

The Zone Information attribute set contains the attributes summarized in Table 8-3.

Table 8-3. Attributes of the Zone Information Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0000	*ZoneState*	enum8	*All*	R	0x00	M
0x0001	*ZoneType*	enum16	*All*	R	-	M
0x0002	*ZoneStatus*	map16	*All*	R	0x00	M

8.2.2.1.1.1 *ZoneState* Attribute

The *ZoneState* attribute contains the values summarized in Table 8-4.

Table 8-4. Values of the *ZoneState* Attribute

Attribute Value	Meaning
0x00	Not enrolled
0x01	Enrolled (the client will react to Zone State Change Notification commands from the server)

8.2.2.1.1.2 *ZoneType* Attribute

The *ZoneType* attribute values are summarized in Table 8-5. The Zone Type dictates the meaning of Alarm1 and Alarm2 bits of the *ZoneStatus* attribute, as also indicated in this table.

Table 8-5. Values of the *ZoneType* Attribute

Value	Zone Type	Alarm1	Alarm2
0x0000	Standard CIE	System Alarm	-
0x000d	Motion sensor	Intrusion indication	Presence indication
0x0015	Contact switch	1st portal Open-Close	2nd portal Open-Close
0x0016	Door/Window handle[117]	See Table 8-7	See Table 8-7
0x0028	Fire sensor	Fire indication	-
0x002a	Water sensor	Water overflow indication	-
0x002b	Carbon Monoxide (CO) sensor	CO indication	Cooking indication
0x002c	Personal emergency device	Fall/Concussion	Emergency button
0x002d	Vibration/Movement sensor	Movement indication	Vibration
0x010f	Remote Control	Panic	Emergency
0x0115	Key fob	Panic	Emergency
0x021d	Keypad	Panic	Emergency
0x0225	Standard Warning Device (see [N1] part 4)	-	-
0x0226	Glass break sensor	Glass breakage detected	-
0x0229	Security repeater*****	-	-
0x8000-0xfffe	manufacturer specific types	-	-
0xffff	Invalid Zone Type	-	-

*** For example: a repeater for security devices that needs to be supervised by the alarm panel/IAS CIE to ensure a reliable security sensor network**

8.2.2.1.1.3 *ZoneStatus* Attribute

The *ZoneStatus* attribute is a bit map. The meaning of each bit is summarized in Table 8-6.

Table 8-6. Values of the *ZoneStatus* Attribute

Bit	Meaning	Values
0	Alarm1	1 – opened or alarmed

[117] NFR Door/Window Position

Bit	Meaning	Values
		0 – closed or not alarmed
1	Alarm2	1 – opened or alarmed 0 – closed or not alarmed
2	Tamper	1 – Tampered 0 – Not tampered
3	Battery	1 – Low battery 0 – Battery OK
4	Supervision Notify	1 – Notify 0 – Does not notify
5	Restore Notify	1 – Notify restore 0 – Does not notify of restore
6	Trouble	1 – Trouble/Failure 0 – OK
7	AC (mains)	1 – AC/Mains fault 0 – AC/Mains OK
8	Test	1 – Sensor is in test mode 0 – Sensor is in operation mode
9	Battery Defect	1 – Sensor detects a defective battery 0 – Sensor battery is functioning normally

For the door/window handle zone type (0x0016), the Alarm1 and Alarm2 bits of the of the *ZoneStatus* attribute are to be interpreted as indicated below.[118]

Table 8-7. Usage of alarm bits of *ZoneStatus* Attribute for *door/window handle* zone type (0x0016)

Alarm1	Alarm2	Description
0b0	0b0	(Door/window) closed
0b1	0b0	(Door/window) tilted (partly open)
0b1	0b1	(Door/window) open

8.2.2.1.1.3.1 Supervision Notify[119]

This bit indicates whether the Zone issues periodic Zone Status Change Notification commands. The CIE device MAY use these periodic notifications as an indication that a zone is operational. Zones that do not implement periodic notifications are required to set this bit to zero (the CIE will know not to interpret the lack of reports as a problem).

[118] NFR Door/Window Position
[119] CCB 2352 explain interval is not defined and clarify that notifications are not general reporting commands

The notification interval is not configurable. It is manufacturer specific and is typically determined by local regulations (e.g. UL requires a minimum of 28 mins). The Poll Control cluster SHOULD be used for sleeping end devices.

8.2.2.1.1.3.2 Restore Notify[120]

This bit indicates whether or not a Zone Status Change Notification command will be sent to indicate that an alarm is no longer present. Some Zones do not have the ability to detect that alarm condition is no longer present, they only can tell that an alarm has occurred. These Zones must set the "Restore" bit to zero, indicating to the CIE not to look for alarm-restore notifications.

8.2.2.1.2 Zone Settings Attribute Set

The Zone Settings attribute set contains the attributes summarized in Table 8-8.

Table 8-8. Attributes of the Zone Settings Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0010	*IAS_CIE_Address*	EUI64	-	RW	-	M
0x0011	*ZoneID*	uint8	0x00 – 0xFF	R	0xFF	M
0x0012	*NumberOfZoneSensitivityLevelsSupported*	uint8	0x02 – 0xff	R	0x02	O*
0x0013	*CurrentZoneSensitivityLevel*	uint8	0x00 – 0xff	RW	0x00	O*

** These attributes depend on each other and if one is supported than both SHALL be supported.*

8.2.2.1.2.1 *IAS_CIE_Address* Attribute

The *IAS_CIE_Address* attribute specifies the address that commands generated by the server SHALL be sent to. All commands received by the server must also come from this address.

It is up to the zone's specific implementation to permit or deny change (write) of this attribute at specific times.

See section 8.2.2.1.3 for more information on setting this attribute.

8.2.2.1.2.2 ZoneID Attribute

A unique reference number allocated by the CIE at zone enrollment time.

Used by IAS devices to reference specific zones when communicating with the CIE. The *ZoneID* of each zone stays fixed until that zone is unenrolled.

8.2.2.1.2.3 NumberOfZoneSensitivityLevelsSupported Attribute

Provides the total number of sensitivity levels supported by the IAS Zone server. The purpose of this attribute is to support devices that can be configured to be more or less sensitive (e.g., motion sensor). It provides IAS Zone clients with the range of sensitivity levels that are supported so they MAY be presented to the user for configuration.

The values 0x00 and 0x01 are reserved because a device that has zero or one sensitivity level SHOULD NOT support this attribute because no configuration of the IAS Zone server's sensitivity level is possible.

The meaning of each sensitivity level is manufacturer-specific. However, the sensitivity level of the IAS Zone server SHALL become more sensitive as they ascend. For example, if the server supports three sensitivity levels, then the value of this attribute would be 0x03 where 0x03 is more sensitive than 0x02, which is more sensitive than 0x01.

8.2.2.1.2.4 CurrentZoneSensitivityLevel Attribute

[120] CCB 2352 change from notes to body text

Allows an IAS Zone client to query and configure the IAS Zone server's sensitivity level. Please see Section *NumberOfZoneSensitivityLevelsSupported* Attribute for more detail on how to interpret this attribute.

The default value 0x00 is the device's default sensitivity level as configured by the manufacturer. It MAY correspond to the same sensitivity as another value in the *NumberOfZoneSensitivityLevelsSupported*, but this is the default sensitivity to be used if the *CurrentZoneSensitivityLevel* attribute is not otherwise configured by an IAS Zone client.

8.2.2.1.3 Implementation Guidelines

Use of the *IAS_CIE_Address* and *ZoneID* attributes functions as an additional enrollment step that is not employed by other devices. The reason for this is to provide an extra layer of security due to the nature of these devices in protecting premises from physical intrusion and attack.

There are three methods for enrolling IAS Zone server to an IAS CIE (i.e., IAS Zone client):

- Trip-to-pair
- Auto-Enroll-Response
- Auto-Enroll-Request

IAS Zone servers SHALL support either:

- Trip-to-pair AND Auto-Enroll-Response, OR
- Auto-Enroll-Request

An IAS Zone client SHALL support either:

- Trip-to-pair AND Auto-Enroll-Response, OR
- Auto-Enroll-Request

An IAS Zone client MAY support all enrollment methods. The Trip-to-Pair enrollment method is primarily intended to be used when there is a desire for an explicit enrollment method (e.g., when a GUI wizard or other commissioning tool is used by a user or installer to add multiple IAS Zone servers in an orderly fashion, assign names to them, configure them in the system).

A commissioning tool MAY act as an agent, on behalf of an IAS CIE device for either commissioning method. A commissioning tool MAY perform any of the actions that are defined below for the IAS CIE.

The following requirements are intended to ensure a timely and interoperable commissioning process:

- After joining a network, an IAS Zone server implemented as a Sleepy End Device SHALL data poll at least once every seven seconds until its *ZoneState* attribute has been updated to "enrolled" (i.e., until it receives a Zone Enroll Response command from an IAS Zone client).
- After joining a network, an IAS Zone server SHOULD data poll at least once every two seconds until its *ZoneState* attribute has been updated to "enrolled" (i.e., until it receives a Zone Enroll Response command from an IAS Zone client).
- If the IAS Zone server supports Poll Control cluster, it SHOULD continue data polling at this rate until its Poll Control cluster parameters are configured otherwise.
- The *IAS_CIE_Address* attribute of the IAS Zone server to be enrolled SHALL be configured only by the IAS CIE (or an agent of an IAS CIE). A self-configuration based on any kind of auto-detect approach triggered by the IAS Zone server itself SHALL be prohibited.

The detailed requirements for each commissioning method follow:

Trip-to-Pair

1. After an IAS Zone server is commissioned to a network, the IAS CIE MAY perform service discovery.
2. If the IAS CIE determines it wants to enroll the IAS Zone server, it SHALL send a Write Attribute command on the IAS Zone server's *IAS_CIE_Address* attribute with its IEEE address.
3. The IAS Zone server MAY configure a binding table entry for the IAS CIE's address because all of its communication will be directed to the IAS CIE.

4. Upon a user input determined by the manufacturer (e.g., a button, change to device's *ZoneStatus* attribute that would result in a Zone Status Change Notification command) and the IAS Zone server's *ZoneState* attribute equal to 0x00 (unenrolled), the IAS Zone server SHALL send a Zone Enroll Request command.
5. The IAS CIE SHALL send a Zone Enroll Response command, which assigns the IAS Zone server's *ZoneID* attribute.
6. The IAS Zone server SHALL change its *ZoneState* attribute to 0x01 (enrolled).

Auto-Enroll-Response

1. After an IAS Zone server is commissioned to a network, the IAS CIE MAY perform service discovery.
2. If the IAS CIE determines it wants to enroll the IAS Zone server, it SHALL send a Write Attribute command on the IAS Zone server's *CIE_IAS_Address* attribute with its IEEE address.
3. The IAS Zone server MAY configure a binding table entry for the IAS CIE's address because all of its communication will be directed to the IAS CIE.
4. The IAS CIE SHALL send a Zone Enroll Response, which assigns the IAS Zone server's *ZoneID* attribute.
5. The IAS Zone server SHALL change its *ZoneState* attribute to 0x01 (enrolled).

Auto-Enroll-Request

1. After an IAS Zone server is commissioned to a network, the IAS CIE MAY perform service discovery.
2. If the IAS CIE determines it wants to enroll the IAS Zone server, it SHALL send a Write Attribute command on the IAS Zone server's *IAS_CIE_Address* attribute with its IEEE address.
3. The IAS Zone server MAY configure a binding table entry for the IAS CIE's address because all of its communication will be directed to the IAS CIE.
4. The IAS Zone server SHALL send a Zone Enroll Request command.
5. The IAS CIE SHALL send a Zone Enroll Response command, which assigns the IAS Zone server's *ZoneID* attribute.
6. The IAS Zone server SHALL change its *ZoneState* attribute to 0x01 (enrolled).

Once the *IAS_CIE_Address* attribute has been written on an IAS Zone server, the IAS Zone server SHALL only act upon commands received from an initiator that matches the *IAS_CIE_Address* attribute.

Any attempt via the ZDO bind or unbind request to create, modify or remove binding table entry on a device embodying the IAS Zone server SHALL be rejected and responded with the status NOT_AUTHORIZED, if the subjected binding table entry is related to an IAS Zone server cluster and the ZDP request does not come from the paired IAS CIE.

8.2.2.2 Commands Received

The command IDs received by the IAS Zone server cluster are listed in Table 8-9Received Command IDs for the IAS Zone Cluster.

Table 8-9. Received Command IDs for the IAS Zone Cluster

Command Id	Description	M/O
0x00	Zone Enroll Response	M
0x01	Initiate Normal Operation Mode	O
0x02	Initiate Test Mode	O

8.2.2.2.1 Zone Enroll Response Command

8.2.2.2.1.1 Payload Format

The Zone Enroll Response command payload SHALL be formatted as illustrated in Figure 8-2Format of the Zone Enroll Response Command Payload.

Figure 8-2. Format of the Zone Enroll Response Command Payload

Bits	8	8
Data Type	enum8	uint8
Field Name	Enroll response code	Zone ID

The permitted values of the Enroll Response Code are shown in Table 8-10 Values of the Enroll Response Code.

Table 8-10. Values of the Enroll Response Code

Code	Meaning	Details
0x00	Success	Success
0x01	Not supported	This specific Zone type is not known to the CIE and is not supported.
0x02	No enroll permit	CIE does not permit new zones to enroll at this time.
0x03	Too many zones	CIE reached its limit of number of enrolled zones

The Zone ID field is the index into the zone table of the CIE (Table 8-12Format of the Zone Table). This field is only relevant if the response code is success.

8.2.2.2.1.2 Effect on Receipt

On receipt, the device embodying the Zone server is notified that it is now enrolled as an active alarm device

The device embodying the Zone server must authenticate received messages by checking the address of their sender against *IAS_CIE_Address*. This is to ensure that only messages from the correct CIE are accepted.

8.2.2.2.2 Initiate Normal Operation Mode Command

Used to tell the IAS Zone server to commence normal operation mode.

8.2.2.2.2.1 Payload Format

This command has no payload.

8.2.2.2.2.2 Effect on Receipt

Upon receipt, the IAS Zone server SHALL commence normal operational mode.

Any configurations and changes made (e.g., *CurrentZoneSensitivityLevel* attribute) to the IAS Zone server SHALL be retained.

Upon commencing normal operation mode, the IAS Zone server SHALL send a Zone Status Change Notification command updating the *ZoneStatus* attribute Test bit to zero (i.e., “operation mode”).

8.2.2.2.2.3 Initiate Test Mode Command

Certain IAS Zone servers MAY have operational configurations that could be configured OTA or locally on the device. This command enables them to be remotely placed into a test mode so that the user or installer MAY configure their field of view, sensitivity, and other operational parameters. They MAY also verify the placement and proper operation of the IAS Zone server, which MAY have been placed in a difficult to reach location (i.e., making a physical input on the device impractical to trigger).

Another use case for this command is large deployments, especially commercial and industrial, where placing the entire IAS system into test mode instead of a single IAS Zone server is infeasible due to the vulnerabilities that might arise. This command enables only a single IAS Zone server to be placed into test mode.

The biggest limitation of this command is that most IAS Zone servers today are battery-powered sleepy nodes that cannot reliably receive commands. However, implementers MAY decide to program an IAS Zone server by factory default to maintain a limited duration of normal polling upon initialization/joining to a new network. Some IAS Zone servers MAY also have AC mains power and are able to receive commands. Some types of IAS Zone servers that MAY benefit from this command are: motion sensors and fire sensor/smoke alarm listeners (i.e., a device that listens for a non-communicating fire sensor to alarm and communicates this to the IAS CIE).

8.2.2.2.2.4 Payload Format

The Initiate Test Mode command SHALL be formatted as illustrated below:

Figure 8-3. Payload format of Initiate Test Mode command

Octets	1	1
Data Type	uint8	uint8
Field Name	Test Mode Duration	Current Zone Sensitivity Level

8.2.2.2.2.5 Test Mode Duration Field

Specifies the duration, in seconds, for which the IAS Zone server SHALL operate in its test mode.

8.2.2.2.2.6 Current Zone Sensitivity Level Field

Specifies the sensitivity level the IAS Zone server SHALL use for the duration of the Test Mode and with which it must update its *CurrentZoneSensitivityLevel* attribute.

The permitted values of Current Zone Sensitivity Level are shown defined for the *CurrentZoneSensitivityLevel* Attribute in Section 8.2.2.1.2.4.

8.2.2.2.2.7 Effect on Receipt

Upon receipt, the IAS Zone server SHALL commence test mode for the duration specified in the command and update its *CurrentZoneSensitivityLevel* attribute to match the value specified in the command.

The IAS Zone server SHALL send a Zone Status Change Notification command updating the *ZoneStatus* attribute Test bit to one (i.e., “test mode”).

While in test mode, the IAS Zone server SHALL send Zone Status Change Notification commands with the appropriate payload to signal that the node is successfully detecting test events.

Upon completing the allotted test mode duration time, the IAS Zone server SHALL resume normal operation mode and SHALL send a Zone Status Change Notification command updating the *ZoneStatus* attribute Test bit to zero (i.e., “normal operation mode”).

At any time, the IAS Zone client MAY send an Initiate Normal Operation Mode command.

The behavior of the IAS Zone server while in test mode is manufacturer specific. The below devices SHOULD behave in the following manner:

- Motion sensor
    - Suspend battery saving features designed to reduce the frequency of motion detection events sent to the IAS CIE.
    - Reduce the "blackout period" (i.e., the period the sensor waits before sending a second motion detection event) so that the IAS Zone server restores an existing motion event and is ready to send another motion event notification within ten seconds. This is sometimes known as "walk test" mode.
    - Support manufacturer specific profile extensions to allow a user to configure additional operational parameters that can be tested while in test mode.
- Fire sensor
    - Begin listening for the user to initiate a test mode on the non-communicating fire sensor or smoke detector
    - Generate a Zone Status Change Notification command upon detecting an audible test alarm
- Carbon monoxide sensor
    - Begin listening for the user to initiate a test mode on the non-communicating carbon monoxide detector
    - Generate a Zone Status Change Notification command upon detecting an audible test alarm

The above guidelines are intended as guidelines for the devices likely to implement test mode. Any IAS Zone server MAY implement test mode features and commands.

Future revisions to this section MAY include additional commands germane to the operational behavior of a given IAS Zone server.

8.2.2.3 Commands Generated

The generated command IDs for the IAS Zone server cluster are listed in Table 8-11 Generated Command IDs for the IAS Zone Cluster.

Table 8-11. Generated Command IDs for the IAS Zone Cluster

Command Identifier	Description	M/O
0x00	Zone Status Change Notification	M
0x01	Zone Enroll Request	M

8.2.2.3.1 Zone Status Change Notification Command

8.2.2.3.1.1 Payload Format

The Zone Status Change Notification command payload SHALL be formatted as illustrated in Figure 8-4Format of the Zone Status Change Notification Command Payload.

Figure 8-4. Format of the Zone Status Change Notification Command Payload

Bits	16	8	8	16
Data Type	map16	map8	uint8	uint16
Field Name	Zone Status	Extended Status	Zone ID	Delay

8.2.2.3.1.2 Zone Status Parameter

The Zone Status field SHALL be the current value of the *ZoneStatus* attribute.

8.2.2.3.1.3 Extended Status Parameter

The Extended Status field is reserved for additional status information and SHALL be set to zero.

8.2.2.3.1.4 Zone ID Parameter

Zone ID is the index of the Zone in the CIE's zone table (Table 8-12). If none is programmed, the Zone ID default value SHALL be indicated in this field.

8.2.2.3.1.5 Delay Parameter

The Delay field is defined as the amount of time, in quarter-seconds, from the moment when a change takes place in one or more bits of the Zone Status attribute and the successful transmission of the Zone Status Change Notification. This is designed to help congested networks or offline servers quantify the amount of time from when an event was detected and when it could be reported to the client.

8.2.2.3.1.6 When Generated

The Zone Status Change Notification command is generated when a change takes place in one or more bits of the *ZoneStatus* attribute.

8.2.2.3.2 Zone Enroll Request Command

8.2.2.3.2.1 Payload Format

The Zone Enroll Request command payload SHALL be formatted as illustrated in Figure 8-5Format of the Zone Enroll Request Command Payload.

Figure 8-5. Format of the Zone Enroll Request Command Payload

Bits	16	16
Data Type	enum16	uint16
Field Name	Zone Type	Manufacturer Code

The Zone Type field SHALL be the current value of the *ZoneType* attribute.

The Manufacturer Code field SHALL be the manufacturer code as held in the node descriptor for the device. See [Z12] Manufacturer Code Database.

8.2.2.3.2.2 When Generated

The Zone Enroll Request command is generated when a device embodying the Zone server cluster wishes to be enrolled as an active alarm device. It must do this immediately it has joined the network (during commissioning).

8.2.3 Client

No dependencies or cluster specific attributes are defined for the client. The client receives the cluster specific commands detailed in 8.2.2.3. The client generates the cluster specific commands detailed in 8.2.2.2, as required by the application.

8.3 IAS ACE

8.3.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The IAS ACE cluster defines an interface to the functionality of any Ancillary Control Equipment of the IAS system. Using this cluster, an ACE device can access a IAS CIE device and manipulate the IAS system, on behalf of a level-2 user (see [N1]).

The client is usually implemented by the IAS ACE device. It allows the IAS ACE device to control the IAS CIE device, which typically implements the server side.

8.3.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added; ZHA 1.2 and 1.2.1 features; CCB 1977

8.3.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	IASACE	Type 1 (client to server)

8.3.1.3 Cluster Identifiers

Identifier	Name
0x0501	IAS ACE

8.3.2 Server

8.3.2.1 Attributes

No attributes are currently defined for this cluster.

8.3.2.2 Zone Table

The Zone Table is used to store information for each Zone enrolled by the CIE. The maximum number of entries in the table is 255.

The format of a group table entry is illustrated in Table 8-12.

Table 8-12. Format of the Zone Table

Field	Type	Valid Range	Description
Zone ID	uint8	0x00 – 0xfe	The unique identifier of the zone
Zone Type	enum16	0x0000 – 0xfffe	See Table 8-5.
Zone Address	EUI64	Valid 64-bit IEEE address	Device address

The Zone ID is a unique reference number allocated by the CIE at zone enrollment time.

The Zone ID is used by IAS devices to reference specific zones when communicating with the CIE. The Zone ID of each zone stays fixed until that zone is un-enrolled.

8.3.2.3 Commands Received

The received command IDs for the IAS ACE server cluster are listed in Table 8-13Received Command IDs for the IAS ACE Cluster.

Table 8-13. Received Command IDs for the IAS ACE Cluster

Command Identifier	Description	M/O
0x00	Arm	M
0x01	Bypass	M
0x02	Emergency	M
0x03	Fire	M
0x04	Panic	M
0x05	Get Zone ID Map	M
0x06	Get Zone Information	M
0x07	Get Panel Status	M
0x08	Get Bypassed Zone List	M
0x09	Get Zone Status	M

8.3.2.3.1 Arm Command

8.3.2.3.1.1 Payload Format

The Arm command payload SHALL be formatted as illustrated in Figure 8-6.

Figure 8-6. Format of the Arm Command Payload

Bits	8	Varies	8
Data Type	enum8	string	uint8
Field Name	Arm Mode	Arm/Disarm Code	Zone ID

8.3.2.3.1.2 Arm Mode Field

The Arm Mode field SHALL have one of the values shown in Table 8-14Arm Mode Field Values.

Table 8-14. Arm Mode Field Values

Value	**Meaning**
0x00	Disarm
0x01	Arm Day/Home Zones Only
0x02	Arm Night/Sleep Zones Only
0x03	Arm All Zones

8.3.2.3.1.3 Arm/Disarm Code Field

The Arm/Disarm Code SHALL be a code entered into the ACE client (e.g., security keypad) or system by the user upon arming/disarming. The server MAY validate the Arm/Disarm Code received from the IAS ACE client in Arm command payload before arming or disarming the system. If the client does not have the capability to input an Arm/Disarm Code (e.g., keyfob), or the system does not require one, the client SHALL a transmit a string with a length of zero.

There is no minimum or maximum length to the Arm/Disarm Code; however, the Arm/Disarm Code SHOULD be between four and eight alphanumeric characters in length.

The string encoding SHALL be UTF-8.

8.3.2.3.1.4 Zone ID Field

Zone ID is the index of the Zone in the CIE's zone table (Table 8-12). If none is programmed, the Zone ID default value SHALL be indicated in this field.

8.3.2.3.1.5 Effect on Receipt

On receipt of this command, the receiving device sets its arm mode according to the value of the Arm Mode field, as detailed in Table 8-14. It is not guaranteed that an Arm command will succeed. Based on the current state of the IAS CIE, and its related devices, the command can be rejected. The device SHALL generate an Arm Response command (see 8.3.2.4.1) to indicate the resulting armed state.

8.3.2.3.2 Bypass Command

Provides IAS ACE clients with a method to send zone bypass requests to the IAS ACE server. Bypassed zones MAY be faulted or in alarm but will not trigger the security system to go into alarm. For example, a user MAY wish to allow certain windows in his premises protected by an IAS Zone server to be left open while the user leaves the premises. The user could bypass the IAS Zone server protecting the window on his IAS ACE client (e.g., security keypad), and if the IAS ACE server indicates that zone is successfully bypassed, arm his security system while he is away.

8.3.2.3.2.1 Payload Format

The Bypass command payload SHALL be formatted as illustrated in Figure 8-7For.

Figure 8-7. Format of the Bypass Command Payload

Bits	8	8	...	8	Varies
Data Type	uint8	uint8	...	uint8	string
Field Name	Number of Zones	Zone ID	...	Zone ID	Arm/Disarm Code

8.3.2.3.2.2 Number of Zones Field

This is the number of Zone IDs included in the payload.

8.3.2.3.2.3 Zone ID Field

Zone ID is the index of the Zone in the CIE's zone table (Table 8-12Format of the Zone Table).

8.3.2.3.2.4 Arm/Disarm Code Field

This field is the same as the Arm/Disarm Code field defined in Section 8.3.2.3.1.3.

8.3.2.3.2.5 Effect of Receipt

On receipt of this command, the IAS ACE server SHALL process this bypass request and generate a single Bypass Response command for the zones requested in the Bypass command payload

8.3.2.3.3 Emergency, Fire and Panic Commands

These commands indicate the emergency situations inherent in their names. They have no payload.

8.3.2.3.4 Get Zone ID Map Command

8.3.2.3.4.1 Payload Format

This command has no payload.

8.3.2.3.4.2 Effect on Receipt

On receipt of this command, the device SHALL generate a Get Zone ID Map Response command. See 8.3.2.4.2.

8.3.2.3.5 Get Zone Information Command

8.3.2.3.5.1 Payload Format

The Get Zone Information command payload SHALL be formatted as illustrated in Figure 8-8.

Figure 8-8. Format of the Get Zone Information Command Payload

Bits	8
Data Type	uint8
Field Name	Zone ID

8.3.2.3.5.2 Effect on Receipt

On receipt of this command, the device SHALL generate a Get Zone Information Response command. See 8.3.2.4.3.

8.3.2.3.6 Get Panel Status Command

This command is used by ACE clients to request an update to the status (e.g., security system arm state) of the ACE server (i.e., the IAS CIE). This command is useful for battery-powered ACE clients with polling rates longer than the standard check-in rate.

8.3.2.3.6.1 Payload Format

There is no payload for the Get Panel Status command.

8.3.2.3.6.2 Effect on Receipt

On receipt of this command, the ACE server responds with the status of the security system. The IAS ACE server SHALL generate a Get Panel Status Response command.

8.3.2.3.7 Get Bypassed Zone List Command

Provides IAS ACE clients with a way to retrieve the list of zones to be bypassed. This provides them with the ability to provide greater local functionality (i.e., at the IAS ACE client) for users to modify the Bypassed Zone List and reduce communications to the IAS ACE server when trying to arm the CIE security system.

8.3.2.3.7.1 Payload Format

This command has no payload.

8.3.2.3.7.2 Effect on Receipt

Upon receipt, the IAS ACE server sends a Set Bypassed Zone List command.

8.3.2.3.8 Get Zone Status Command

This command is used by ACE clients to request an update of the status of the IAS Zone devices managed by the ACE server (i.e., the IAS CIE). This command is useful for battery-powered ACE clients with polling rates longer than the standard check-in rate. The command is similar to the Get Attributes Supported command in that it specifies a starting Zone ID and a number of Zone IDs for which information is requested.

Depending on the number of IAS Zone devices managed by the IAS ACE server, sending the Zone Status of all zones MAY not fit into a single Get Zone Status Response command. IAS ACE clients MAY need to send multiple Get Zone Status commands in order to get the information they seek.

8.3.2.3.8.1 Payload Format

The Get Zone Status command SHALL be formatted as illustrated below.

Figure 8-9. Format of the Get Zone Status command

Bits	8	8	8	16
Data Type	uint8	uint8	bool	map16
Field Name	Starting Zone ID	Max Number of Zone IDs	Zone Status Mask Flag	Zone Status Mask

8.3.2.3.8.2 Starting Zone ID Field

Specifies the starting Zone ID at which the IAS Client would like to obtain zone status information.

8.3.2.3.8.3 Max Number of Zone IDs Requested Field

Specifies the maximum number of Zone IDs and corresponding Zone Statuses that are to be returned by the IAS ACE server when it responds with a Get Zone Status Response command.

8.3.2.3.8.4 Zone Status Mask Flag Field

Functions as a query operand with the Zone Status Mask field. If set to zero (i.e., FALSE), the IAS ACE server SHALL include all Zone IDs and their status, regardless of their Zone Status when it responds with a Get Zone Status Response command.

If set to one (i.e., TRUE), the IAS ACE server SHALL include only those Zone IDs whose *Zone Status* attribute is equal to one or more of the Zone Statuses requested in the Zone Status Mask field of the Get Zone Status command.

Use of Zone Status Mask Flag and Zone Status Mask fields allow a client to obtain updated information for the subset of Zone IDs they're interested in, which is beneficial when the number of IAS Zone devices in a system is large.

8.3.2.3.8.5 Zone Status Mask Field

Coupled with the Zone Status Mask Flag field, functions as a mask to enable IAS ACE clients to get information about the Zone IDs whose *ZoneStatus* attribute is equal to any of the bits indicated by the IAS ACE client in the Zone Status Mask field. The format of this field is the same as the *ZoneStatus* attribute in the IAS Zone cluster. Per the Zone Status Mask Flag field, IAS ACE servers SHALL respond with only the Zone IDs whose *ZoneStatus* attributes are equal to at least one of the Zone Status bits set in the Zone Status Mask field requested by the IAS ACE client.

For example, if the Zone Status Mask field set to "0x0003" would match IAS Zones whose *ZoneStatus* attributes are 0x0001, 0x0002, and 0x0003. In other words, if a logical 'AND' between the Zone Status Mask field and the IAS Zone's *ZoneStatus* attribute yields a non-zero result, the IAS ACE server SHALL include that IAS Zone in the Get Zone Status Response command.

8.3.2.3.8.6 Effect on Receipt

On receipt of this command, the IAS ACE server responds with the status of the zones it manages that meet those requested in the Get Zone Status command. The IAS ACE server SHALL generate a Get Zone Status Response command.

8.3.2.4 Commands Generated

The generated command IDs for the IAS ACE server cluster are listed in Table 8-15.

Table 8-15. Generated Command IDs for the IAS ACE Cluster

Command Identifier	Description	M/O
0x00	Arm Response	M
0x01	Get Zone ID Map Response	M
0x02	Get Zone Information Response	M
0x03	Zone Status Changed	M
0x04	Panel Status Changed	M
0x05	Get Panel Status Response	M
0x06	Set Bypassed Zone List	M
0x07	Bypass Response	M
0x08	Get Zone Status Response	M

8.3.2.4.1 Arm Response Command

8.3.2.4.1.1 Payload Format

The Arm Response command payload SHALL be formatted as illustrated in Figure 8-10.

Figure 8-10. Format of the Arm Response Command Payload

Bits	8
Data Type	enum8
Field Name	Arm Notification

8.3.2.4.1.2 Arm Notification Field

The Arm Notification field SHALL have one of the values shown in Table 8-16.

Table 8-16. Arm Notification Values

Value	Meaning
0x00	All Zones Disarmed
0x01	Only Day/Home Zones Armed
0x02	Only Night/Sleep Zones Armed
0x03	All Zones Armed
0x04	Invalid Arm/Disarm Code

Value	Meaning
0x05	Not ready to arm*****
0x06	Already disarmed

* NOTE: reasons for not being ready to arm are determined by the IAS ACE server manufacturer

8.3.2.4.2 Get Zone ID Map Response Command

8.3.2.4.2.1 Payload Format

The Get Zone ID Map Response command payload SHALL be formatted as illustrated in Figure 8-11.

Figure 8-11. Get Zone ID Map Response Command Payload

Bits	16	. . .	16
Data Type	map16	. . .	map16
Field Name	Zone ID Map section 0	. . .	Zone ID Map section 15

The 16 fields of the payload indicate whether each of the Zone IDs from 0 to 0xff is allocated or not. If bit n of Zone ID Map section N is set to 1, then Zone ID (16 x N + n) is allocated, else it is not allocated.

8.3.2.4.3 Get Zone Information Response Command

8.3.2.4.3.1 Payload Format

The Get Zone Information Response command payload SHALL be formatted as illustrated in Figure 8-12.

Figure 8-12. Format of the Get Zone Information Response Command Payload

Bits	8	16	64	Varies
Data Type	uint8	enum16	EUI64	string
Field Name	Zone ID	Zone Type	IEEE address	Zone Label

The first 3 fields of the payload are equal to the fields of the Zone Table entry corresponding to the ZoneID field of the Get Zone Information command to which this command is a response.

If the Zone ID is unallocated, this SHALL be indicated by setting the Zone Type and IEEE Address fields to 0xffff (see Table 8-5Values of the) and 0xffffffffffffffff respectively.

8.3.2.4.3.2 Zone Label Field

Provides the Zone Label stored in the IAS CIE. If none is programmed, the IAS ACE server SHALL transmit a string with a length of zero.

There is no minimum or maximum length to the Zone Label field; however, the Zone Label SHOULD be between 16 to 24 alphanumeric characters in length.

The string encoding SHALL be UTF-8.

8.3.2.4.4 Zone Status Changed Command

This command updates ACE clients in the system of changes to zone status recorded by the ACE server (e.g., IAS CIE device).

An IAS ACE server SHOULD send a Zone Status Changed command upon a change to an IAS Zone device's *ZoneStatus* that it manages (i.e., IAS ACE server SHOULD send a Zone Status Changed command upon receipt of a Zone Status Change Notification command).

8.3.2.4.4.1 Payload Format

The Zone Status Changed Command SHALL be formatted as illustrated in Figure 8-13.

Figure 8-13. Format of the Zone Status Changed Command Payload

Bits	8	16	8	Varies
Data Type	uint8	enum16	enum8	string
Field Name	Zone ID	Zone Status	Audible Notification	Zone Label

8.3.2.4.4.2 Zone ID Field

The index of the Zone in the CIE's zone table (Table 8-12). If none is programmed, the ZoneID attribute default value SHALL be indicated in this field.

8.3.2.4.4.3 Zone Status Field

The current value of the ZoneStatus attribute.

8.3.2.4.4.4 Audible Notification Field

Provide the ACE client with information on which type of audible notification it SHOULD make for the zone status change. This field is useful for telling the ACE client to play a standard chime or other audio indication or to mute and not sound an audible notification at all. This field also allows manufacturers to create additional audible alert types (e.g., dog barking, windchimes, conga drums) to enable users to customize their system.

The Audible Notification field SHALL be formatted as illustrated below:

Figure 8-14. Audible Notification field value

Enumeration	Description
0x00	Mute (i.e., no audible notification)
0x01	Default sound
0x80-0xff	Manufacturer specific

The default value SHALL be 0x01.

8.3.2.4.4.5 Zone Label Field

Provides the Zone Label stored in the IAS CIE. If none is programmed, the IAS ACE server SHALL transmit a string with a length of zero.

There is no minimum or maximum length to the Zone Label field; however, the Zone Label SHOULD be between 16 to 24 alphanumeric characters in length.

The string encoding SHALL be UTF-8.

8.3.2.4.5 Panel Status Changed Command

This command updates ACE clients in the system of changes to panel status recorded by the ACE server (e.g., IAS CIE device).

Sending the Panel Status Changed command (vs. the Get Panel Status and Get Panel Status Response method) is generally useful only when there are IAS ACE clients that data poll within the retry timeout of the network (e.g., less than 7.68 seconds).

An IAS ACE server SHALL send a Panel Status Changed command upon a change to the IAS CIE’s panel status (e.g., Disarmed to Arming Away/Stay/Night, Arming Away/Stay/Night to Armed, Armed to Disarmed) as defined in the Panel Status field.

When Panel Status is Arming Away/Stay/Night, an IAS ACE server SHOULD send Panel Status Changed commands every second in order to update the Seconds Remaining. In some markets (e.g., North America), the final 10 seconds of the Arming Away/Stay/Night sequence requires a separate audible notification (e.g., a double tone).

8.3.2.4.5.1 Payload Format

The Panel Status Changed Command SHALL be formatted as illustrated in Figure 8-15.

Figure 8-15. Format of the Panel Status Changed Command Payload

Bits	8	8	8	8
Data Type	enum8	uint8	enum8	enum8
Field Name	Panel Status	Seconds Remaining	Audible Notification	Alarm Status

8.3.2.4.5.2 PanelStatus Parameter

The Panel Status parameter SHALL be formatted as illustrated in Table 8-17.

Table 8-17. *PanelStatus* Field Values

Panel Status Enumerations	**Description**
0x00	Panel disarmed (all zones disarmed) and ready to arm
0x01	Armed stay
0x02	Armed night
0x03	Armed away
0x04	Exit delay
0x05	Entry delay
0x06	Not ready to arm

Panel Status Enumerations	Description
0x07	In alarm
0x08	Arming Stay
0x09	Arming Night
0x0a	Arming Away

8.3.2.4.5.3 Audible Notification Field

See 8.3.2.4.4.4 for a description of this field.

8.3.2.4.5.4 Alarm Status Field

Provides the ACE client with information on the type of alarm the panel is in if its Panel Status field indicates it is "in alarm." This field MAY be useful for ACE clients to display or otherwise initiate notification for users.

The Alarm Status field SHALL be formatted as illustrated below. Note: this is the same as the Warning Mode field in the IAS WD cluster.

Figure 8-16. Alarm Status field value

Enumeration	Description
0x00	No alarm
0x01	Burglar
0x02	Fire
0x03	Emergency
0x04	Police Panic
0x05	Fire Panic
0x06	Emergency Panic (i.e., medical issue)

The default value SHALL be 0x00.

8.3.2.4.5.5 Seconds Remaining Parameter

Indicates the number of seconds remaining for the server to be in the state indicated in the Panel Status parameter.

The Seconds Remaining parameter SHALL be provided if the Panel Status parameter has a value of 0x04 (Exit delay) or 0x05 (Entry delay).

The default value SHALL be 0x00.

8.3.2.4.6 Get Panel Status Response Command

This command updates requesting IAS ACE clients in the system of changes to the security panel status recorded by the ACE server (e.g., IAS CIE device).

8.3.2.4.6.1 Payload Format

The Get Panel Status Response command SHALL be formatted as illustrated below.

Figure 8-17. Get Panel Status Response command

Bits	8	8	8	8
Data Type	enum8	uint8	enum8	enum8
Field Name	Panel Status	Seconds Remaining	Audible Notification	Alarm Status

8.3.2.4.6.2 Panel Status Field

See 8.3.2.4.5.2 for a description of this field.

8.3.2.4.6.3 Seconds Remaining Field

Indicates the number of seconds remaining for the server to be in the state indicated in the Panel Status field. The Seconds Remaining field SHALL be provided if the Panel Status field has a value of 0x04 (Exit delay) or 0x05 (Entry delay).

The default value SHALL be 0x00.

8.3.2.4.6.4 Audible Notification Field

See 8.3.2.4.4.4 for a description of this field.

8.3.2.4.6.5 Alarm Status Field

See 8.3.2.4.5.4 for a description of this field.

8.3.2.4.7 Set Bypassed Zone List Command

Sets the list of bypassed zones on the IAS ACE client. This command can be sent either as a response to the Get Bypassed Zone List command or unsolicited when the list of bypassed zones changes on the ACE server.

8.3.2.4.7.1 Payload Format

The Set Bypassed Zone List command SHALL be formatted as illustrated below.

Figure 8-18. Set Bypassed Zone List Command payload format

Bits	8	8	8	…	8
Data Type	uint8	uint8	uint8	…	uint8

Field Name	Number of Zones	Zone ID 1	Zone ID 2	…	Zone ID *n*

8.3.2.4.7.2 Number of Zones Field

This is the number of Zone IDs included in the payload.

If no zones are bypassed, the IAS ACE server SHALL send the Set Bypassed Zone List command with a Number of Zones field set to “0” (zero).

8.3.2.4.7.3 Zone ID 1…X Field

Zone ID is the index of the Zone in the CIE's zone table and is an array of Zone IDs for each zone that is bypassed where X is equal to the value of the Number of Zones field. There is no order imposed by the numbering of the Zone ID field in this command payload. IAS ACE servers SHOULD provide the array of Zone IDs in ascending order.

8.3.2.4.7.4 Implementation Guidelines

The IAS ACE server SHALL reset (i.e., set to be “not bypassed”) all previously bypassed zones each time the security system is disarmed unless certain zones are programmed by the user to be bypassed on a permanent basis.

8.3.2.4.8 Bypass Response Command

Provides the response of the security panel to the request from the IAS ACE client to bypass zones via a Bypass command.

8.3.2.4.8.1 Payload Format

The Bypass Response command SHALL be formatted as illustrated below:

Figure 8-19. Bypass Response command format

Bits	8	8	8	…	8
Data Type	uint8	uint8	uint8	…	uint8
Field Name	Number of Zones	Bypass Result for Zone ID 1	Bypass Result for Zone ID 2	…	Bypass Result for Zone ID *n*

8.3.2.4.8.2 Number of Zones Field

This is the number of Zone IDs for which a bypass result is provided in the payload.

8.3.2.4.8.3 Bypass Result Field

An array of Zone IDs for each zone requested to be bypassed via the Bypass command where X is equal to the value of the Number of Zones field. The order of results for Zone IDs SHALL be the same as the order of Zone IDs sent in the Bypass command by the IAS ACE client.

The permitted values of Bypass Result are shown below:

Table 8-18. Values of Bypass Result Field

Value	Meaning	Description
0x00	Zone bypassed	The Zone ID requested to be bypassed is successful. Zone is bypassed.
0x01	Zone not bypassed	The Zone ID requested to be bypassed is unsuccessful. Zone is not bypassed.
0x02	Not allowed	The Zone ID requested to be bypassed is not eligible to be bypassed per the policy or user configurations on the IAS ACE server. Zone is not bypassed.
0x03	Invalid Zone ID	The Zone ID requested to be bypassed is not in the valid range of Zone IDs.
0x04	Unknown Zone ID	The Zone ID requested to be bypassed is in the valid range of Zone IDs, but the IAS ACE server does not have a record of the Zone ID requested.
0x05	Invalid Arm/Disarm Code	A value returned indicating that the Arm/Disarm Code was entered incorrectly.

8.3.2.4.9 Get Zone Status Response Command

This command updates requesting IAS ACE clients in the system of changes to the IAS Zone server statuses recorded by the ACE server (e.g., IAS CIE device).

8.3.2.4.9.1 Payload Format

The Get Zone Status Response command SHALL be formatted as illustrated below.

Figure 8-20. Format of the Get Zone Status Response command

Bits	8	8	8	16
Data Type	Boolean	uint8	uint8	map16
Field Name	Zone Status Complete	Number of Zones	Zone ID 1	Zone ID 1 Zone Status

Bits	8	16	…	8	16
Data Type	uint8	map16	…	uint8	map16
Field Name	Zone ID 2	Zone ID 2 Zone Status	…	Zone ID N	Zone ID *n* Zone Status

8.3.2.4.9.2 Zone Status Complete Field

Indicates whether there are additional Zone IDs managed by the IAS ACE Server with Zone Status information to be obtained. A value of zero (i.e., FALSE) indicates there are additional Zone IDs for which Zone Status information is available and that the IAS ACE client SHOULD send another Get Zone Status command.

A value of one (i.e., TRUE) indicates there are no more Zone IDs for the IAS ACE client to query and the IAS ACE client has received all the Zone Status information for all IAS Zones managed by the IAS ACE server. The IAS ACE client SHOULD NOT typically send another Get Zone Status command.

8.3.2.4.9.3 Number of Zones Field

This is the number of Zone IDs for which a zone status result is provided in the payload.

8.3.2.4.9.4 Zone ID Field

The index of the Zone in the CIE's zone table. If none is programmed, the *ZoneID* attribute default value SHALL be indicated in this field.

8.3.2.4.9.5 Zone Status Field

The current value of the *ZoneStatus* attribute for the indicated Zone ID.

8.3.3 Client

The client supports no cluster specific attributes. The client receives the cluster specific commands detailed in 8.3.2.4.The client cluster generates the commands detailed in 8.3.2.3, as required by the application.

8.4 IAS WD

8.4.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The IAS WD cluster provides an interface to the functionality of any Warning Device equipment of the IAS system. Using this cluster, a CIE device can access an IAS WD device and issue alarm warning indications (siren, strobe lighting, etc.) when a system alarm condition is detected (according to [N1]).

8.4.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added
2	CCB 2350 2341

8.4.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	IASWD	Type 1 (client to server)

8.4.1.3 Cluster Identifiers

Identifier	Name
0x0502	IAS WD

8.4.2 Server

8.4.2.1 Attributes

The attributes defined for the server cluster are detailed in Table 8-19.

Table 8-19. Attributes of the IAS WD (Server) Cluster

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*MaxDuration*	uint16	0x0000 – 0fffe	RW	240	M

8.4.2.1.1 *MaxDuration* Attribute

The *MaxDuration* attribute specifies the maximum time in seconds that the siren will sound continuously, regardless of start/stop commands.

8.4.2.2 Commands Received

The received command IDs are listed in Table 8-20.

Table 8-20. Received Command IDs for the IAS WD Server Cluster

Command Identifier	Description	M/O
0x00	Start warning	M
0x01	Squawk	M

8.4.2.2.1 Start Warning Command

This command starts the WD operation. The WD alerts the surrounding area by audible (siren) and visual (strobe) signals.

A Start Warning command SHALL always terminate the effect of any previous IAS WD cluster command that is still current.

8.4.2.2.1.1 Payload Format

The Start Warning command payload SHALL be formatted as illustrated in Figure 8-21.

Figure 8-21. Format of the Start Siren Command Payload

Bits	4	2	2	16	8	8
Data Type	map8			uint16	uint8	enum8
Field Name	Warning Mode	Strobe	Siren Level	Warning Duration	Strobe Duty Cycle	Strobe Level

The Warning Mode and Strobe subfields are concatenated together to a single 8-bit bitmap field. The groups of bits these subfields occupy are used as described below.

8.4.2.2.1.2 Warning Mode Field

The Warning Mode field is used as an 4-bit enumeration, can have one of the values defined below. The exact behavior of the WD device in each mode is according to the relevant security standards.

Table 8-21. Warning Modes

Warning Mode	Meaning
0	Stop (no warning)
1	Burglar
2	Fire
3	Emergency
4	Police panic
5	Fire panic
6	Emergency Panic (i.e., medical issue)

8.4.2.2.1.3 Strobe Field

The Strobe field is used as a 2-bit enumeration, and determines if the visual indication is required in addition to the audible siren, as indicated in Table 8-22. If the strobe field is “1” and the Warning Mode is “0” (“Stop”) then only the strobe is activated.

Table 8-22. Values of the Strobe Field

Value	Meaning
0	No strobe
1	Use strobe in parallel to warning

8.4.2.2.1.4 Siren Level Field

The Siren Level field is used as a 2-bit enumeration, and indicates the intensity of audible squawk sound as shown in the following table. At least one level of sound SHALL be supported[121].

Table 8-23. Siren Level Field Values

***SirenLevel* Value**	**Description**
0	Low level sound
1	Medium level sound
2	High level sound
3	Very high level sound

8.4.2.2.1.5 Warning Duration Field

Requested duration of warning, in seconds. If both Strobe and Warning Mode are "0" this field SHALL be ignored.

8.4.2.2.1.6 Strobe Duty Cycle Field

Indicates the length of the flash cycle. This provides a means of varying the flash duration for different alarm types (e.g., fire, police, burglar). Valid range is 0-100 in increments of 10. All other values SHALL be rounded to the nearest valid value. Strobe SHALL calculate duty cycle over a duration of one second. The ON state SHALL precede the OFF state. For example, if Strobe Duty Cycle Field specifies “40,” then the strobe SHALL flash ON for 4/10ths of a second and then turn OFF for 6/10ths of a second.

The default value for this field SHALL be 0x00.

8.4.2.2.1.7 Strobe Level Field

Indicates the intensity of the strobe as shown in the table below. This attribute is designed to vary the output of the strobe (i.e., brightness) and not its frequency, which is detailed in 8.4.2.2.1.6. At least one level of strobe SHALL be supported[122].

Table 8-24. Strobe Level Field Values

***StrobeLevel* Enumerations**	**Description**
0x00	Low level strobe
0x01	Medium level strobe
0x02	High level strobe
0x03	Very high level strobe

8.4.2.2.2 Squawk Command

This command uses the WD capabilities to emit a quick audible/visible pulse called a "squawk". The squawk command has no effect if the WD is currently active (warning in progress).

8.4.2.2.2.1 Payload Format

[121] CCB 2350 2341 clarify that only one level need be supported
[122] CCB 2350 2341 clarify that only one level need be supported

The Squawk command payload SHALL be formatted as illustrated in Figure 8-22.

Figure 8-22. Format of the Start Siren Command Payload

Bits	4	1	1	2
Data Type	map8			
Field Name	Squawk mode	Strobe	Reserved	Squawk level

8.4.2.2.2.2 Squawk Mode Field

The Squawk Mode field is used as a 4-bit enumeration, and can have one of the values shown in Table 8-25 Squawk Mode Field. The exact operation of each mode (how the WD “squawks”) is implementation specific.

Table 8-25. Squawk Mode Field

Warning Mode	**Meaning**
0	Notification sound for “System is armed”
1	Notification sound for "System is disarmed"

8.4.2.2.2.3 Strobe Field

The strobe field is used as a Boolean, and determines if the visual indication is also required in addition to the audible squawk, as shown in Table 8-26 Strobe Bit.

Table 8-26. Strobe Bit

Value	**Meaning**
0	No strobe
1	Use strobe blink in parallel to squawk

8.4.2.2.2.4 Squawk Level Field

The squawk level field is used as a 2-bit enumeration, and determines the intensity of audible squawk sound as shown in Table 8-27 Squawk Level Field Values.

Table 8-27. Squawk Level Field Values

Value	**Meaning**
0	Low level sound
1	Medium level sound
2	High level sound
3	Very High level sound

8.4.2.3 Commands Generated

No cluster specific commands are generated by the server cluster.

8.4.3 Client

The client side is implemented by the CIE. The CIE is a client of the warning service provided by this cluster. Usually a WD would implement an IAS WD cluster server and an IAS Zone cluster server.

There are no cluster specific attributes defined for the client cluster. The client receives no cluster specific commands. The client cluster generates the cluster specific commands detailed in 8.4.2.2, as required by the application.

CHAPTER 9 PROTOCOL INTERFACES

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

9.1 General Description

9.1.1 Introduction

The clusters specified in this document are for use in applications which interface to external protocols.

9.1.2 Cluster List

This section lists the clusters specified in this document, and gives examples of typical usage for the purpose of clarification.

The clusters defined in this document are listed in Table 9-1.

Table 9-1. Clusters of the Protocol Interfaces Functional

Cluster ID	Cluster Name	Description
0x0016	Partition	The commands and attributes for enabling partitioning of a large frame between devices
0x0600	Generic tunnel	The minimum common commands and attributes required to tunnel any protocol.
0x0601	BACnet protocol tunnel	Commands and attributes required to tunnel the BACnet protocol.
0x0602	Analog input (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of an analog measurement.
0x0603	Analog input (BACnet extended)	An interface for accessing a number of BACnet based attributes of an analog measurement.
0x0604	Analog output (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of an analog output.
0x0605	Analog output (BACnet extended)	An interface for accessing a number of BACnet based attributes of an analog output.
0x0606	Analog value(BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of an analog value, typically used as a control system parameter.
0x0607	Analog value(BACnet extended)	An interface for accessing a number of BACnet based attributes of an analog value, typically used as a control system parameter.

Cluster ID	Cluster Name	Description
0x0608	Binary input (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a binary measurement.
0x0609	Binary input (BACnet extended)	An interface for accessing a number of BACnet based attributes of a binary measurement.
0x060a	Binary output (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a binary output.
0x060b	Binary output (BACnet extended)	An interface for accessing a number of BACnet based attributes of a binary output.
0x060c	Binary value (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a binary value, typically used as a control system parameter.
0x060d	Binary value (BACnet extended)	An interface for accessing a number of BACnet based attributes of a binary value, typically used as a control system parameter.
0x060e	Multistate input (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a multistate measurement.
0x060f	Multistate input (BACnet extended)	An interface for accessing a number of BACnet based attributes of a multistate measurement.
0x0610	Multistate output (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a multistate output.
0x0611	Multistate output (BACnet extended)	An interface for accessing a number of BACnet based attributes of a multistate output.
0x0612	Multistate value (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a multistate value, typically used as a control system parameter.
0x0613	Multistate value (BACnet extended)	An interface for accessing a number of BACnet based attributes of a multistate value, typically used as a control system parameter.
0x0614	11073 Protocol Tunnel	Interface for 11073 Protocol Tunnel used in health care applications
0x0615	ISO7816 Tunnel	Commands and attributes for mobile office solutions using devices.

9.2 Generic Tunnel

9.2.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The generic cluster provides the minimum common commands and attributes required to discover protocol tunnelling devices. A protocol cluster specific to the protocol being tunneled shall be implemented on the same endpoint as the Generic Tunnel cluster.

Note: The reverse is not true, as there may be tunnel clusters that do not require the Generic Tunnel cluster.

9.2.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	TUN	Type 1 (client to server)

9.2.1.3 Cluster Identifiers

Identifier	Name
0x0600	Generic Tunnel

9.2.2 Server

9.2.2.1 Dependencies

The maximum size of the *ProtocolAddress* attribute is dependent on the protocol supported by any associated specific protocol tunnel cluster supported on the same endpoint (see 9.2.2.2.3, ProtocolAddress Attribute).

9.2.2.2 Attributes

The Generic Tunnel contains the attributes summarized in Table 9-2.

Table 9-2. Attributes of the Generic Tunnel Cluster

Id	Name	Type	Range	Acc	Default	M/O
0x0001	MaximumIncomingTransferSize	uint16	0x0000 - 0xffff	R	0x0000	M
0x0002	MaximumOutgoingTransferSize	uint16	0x0000 - 0xffff	R	0x0000	M
0x0003	ProtocolAddress	octstr	0 - 255 octets	RW	Null string	M

9.2.2.2.1 MaximumIncomingTransferSize Attribute

The *MaximumIncomingTransferSize* attribute specifies the maximum size, in octets, of the application service data unit (ASDU) that can be transferred to this node in one single message transfer. The ASDU referred to is the ZCL frame, including header and payload, of any command received by a protocol specific tunnel cluster on the same endpoint.

This value cannot exceed the Maximum Incoming Transfer Size field of the node descriptor on the device supporting this cluster.

9.2.2.2.2 MaximumOutgoingTransferSize Attribute

The *MaximumOutgoingTransferSize* attribute specifies the maximum size, in octets, of the application sub-layer data unit (ASDU) that can be transferred from this node in one single message transfer. The ASDU referred to is the ZCL frame, including header and payload, of any command sent by a protocol specific tunnel cluster on the same endpoint.

This value cannot exceed the Maximum Outgoing Transfer Size field of the node descriptor on the device supporting this cluster.

9.2.2.2.3 ProtocolAddress Attribute

The *ProtocolAddress* attribute contains an octet string that is interpreted as a device address by the protocol being tunneled by an associated protocol specific tunnel cluster (if any). The overall maximum size of the string is 255 octets, but devices need only support the actual maximum size required by that protocol

9.2.2.3 Commands Received

The cluster specific commands received by the Generic Tunnel server cluster are listed in Table 9-3.

Table 9-3. Command IDs Received by the Generic Tunnel Cluster

Identifier	Description	M/O
0x00	Match Protocol Address	M

9.2.2.3.1 Match Protocol Address Command

The Match Protocol Address command payload shall be formatted as illustrated in Figure 9-1.

Figure 9-1. Format of Match Protocol Address Command Payload

octets	Variable
Data Type	octstr
Field Name	Protocol Address

9.2.2.3.2 When Generated

This command is generated when an associated protocol specific tunnel cluster wishes to find the address (node, endpoint) of the Generic Tunnel server cluster representing a protocol-specific device with a given protocol address. The command is typically multicast to a group of inter-communicating Generic Tunnel clusters.

9.2.2.3.3 Effect on Receipt

On receipt of this command, a device shall match the Protocol Address field of the received command to the ProtocolAddress attribute. If they are equal, it shall return the Match Protocol Address Response command (see 9.2.2.4.1), otherwise it shall do nothing.

9.2.2.4 Commands Generated

The cluster specific commands generated by the Generic Tunnel server cluster are listed in Table 9-4. Command IDs Generated by the Generic Tunnel Cluster.

Table 9-4. Command IDs Generated by the Generic Tunnel Cluster

Identifier	Description	M/O
0x00	Match Protocol Address Response	M
0x01	Advertise Protocol Address	O

9.2.2.4.1 Match Protocol Address Response Command

The Match Protocol Address Response command payload shall be formatted as illustrated in Figure 9-2.

Figure 9-2. Match Protocol Address Response Command Payload

octets	8	Variable
Data Type	EUI64	octstr
Field Name	Device IEEE Address	Protocol Address

The Device IEEE Address field shall be set equal to the IEEE address of the responding device. The Protocol Address field shall be set equal to the matched Protocol Address.

9.2.2.4.2 When Generated

This command is generated upon receipt of a Match Protocol Address command (see 9.2.2.3.1), to indicate that the Protocol Address was successfully matched by the responding device.

9.2.2.4.3 Advertise Protocol Address Command

The Advertise Protocol Address command payload shall be formatted as illustrated in Figure 9-3.

Figure 9-3. Advertise Protocol Address Command Payload

octets	Variable
Data Type	octstr
Field Name	Protocol Address

The Protocol Address field shall be set to the value of the *ProtocolAddress* attribute.

9.2.2.4.4 When Generated

This command is typically sent upon startup, and whenever the *ProtocolAddress* attribute changes. It is typically multicast to a group of inter-communicating Generic Tunnel clusters.

9.2.3 Client

The client cluster has no specific attributes or dependencies. The client cluster receives the cluster specific commands detailed in Commands Generated. The client cluster generates the cluster specific commands detailed in 9.2.2.3.

9.3 BACnet Protocol Tunnel

9.3.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The BACnet Protocol Tunnel cluster provides the commands and attributes required to tunnel the BACnet protocol (see [A1]). The server cluster receives BACnet NPDUs and the client cluster generates BACnet NPDUs, thus it is necessary to have both server and client on an endpoint to tunnel BACnet messages in both directions.

9.3.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.3.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BACTUN	Type 1 (client to server)

9.3.1.3 Cluster Identifiers

Identifier	Name
0x0601	BACnet Protocol Tunnel

9.3.2 Server

9.3.2.1 Dependencies

Any endpoint that supports the BACnet Protocol Tunnel server cluster shall also support the Generic Tunnel server cluster.

The associated Generic Tunnel server cluster shall have its *ProtocolAddress* attribute equal to the device identifier of the BACnet device represented on that endpoint, expressed as an octet string (i.e., with identical format as a BACnet OID data type, but interpreted as an octet string). The special three octet value 0x3FFFFF of the *ProtocolAddress* attribute indicates that the associated BACnet device is not commissioned.

The associated Generic Tunnel server cluster shall also have its *MaximumIncomingTransferSize* attribute and *MaximumOutgoingTransferSize* attribute equal to or greater than 504 octets. Accordingly, this cluster requires fragmentation to be implemented, with maximum transfer sizes given by these attributes.

9.3.2.2 Attributes

The BACnet Protocol Tunnel cluster does not contain any attributes.

9.3.2.3 Commands Received

The cluster specific commands received by the BACnet Protocol Tunnel server cluster are listed in Table 9-5.

Table 9-5. Command IDs for the BACnet Protocol Tunnel Cluster

Identifier	Description	M/O
0x00	Transfer NPDU	M

9.3.2.3.1 Transfer NPDU Command

9.3.2.3.1.1 Payload Format

The Transfer NPDU command payload shall be formatted as illustrated in Figure 9-4.

Figure 9-4. Format of the Transfer NPDU Command Payload

octets	Variable
Data Type	Sequence of data8
Field Name	NPDU

9.3.2.3.1.2 NPDU Field

The NPDU field is of variable length and is a BACnet NPDU as defined in the BACnet standard [A1]. Its format is a sequence of 8-bit data (see General Data section of Chapter 2 of arbitrary length.

9.3.2.3.1.3 When Generated

This command is generated when a BACnet network layer wishes to transfer a BACnet NPDU across a tunnel to another BACnet network layer.

9.3.2.3.1.4 Effect on Receipt

On receipt of this command, a device shall process the BACnet NPDU as specified in the BACnet standard [A1].

9.3.2.4 Commands Generated

No cluster specific commands are generated by the server cluster.

9.3.3 Client

The client cluster has no specific attributes or dependencies. The client does not receive any cluster specific commands. The cluster specific commands generated by the client cluster are listed in 9.3.2.3.

9.4 BACnet Input, Output and Value Clusters

9.4.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This section specifies a number of clusters which are based on the Input, Output and Value objects specified by BACnet (see [A1]).

Each of these three objects is specified by BACnet in three different forms - Analog, Binary and Multistate. clusters are specified here based on all nine such BACnet objects.

Each such BACnet object is represented in the ZCL by three related clusters – a BACnet Basic cluster , a BACnet Regular cluster and a BACnet Extended cluster. The properties of each BACnet object are implemented as attributes, and are divided into three sets, which are allocated to the clusters as follows.

BACnet Basic clusters implement attributes and functionality that can be readily employed either via interworking with a BACnet system, or by a non-BACnet system. Accordingly, these clusters are included in the ZCL General functional domain.

BACnet Regular and BACnet Extended clusters implement attributes and functionality that are specifically intended for interworking with a BACnet system (through a BACnet gateway). Accordingly, these clusters are included in the ZCL Protocol Interface functional domain.

A BACnet Regular cluster may only be implemented on an endpoint that also implements its associated Basic cluster. Similarly, a BACnet Extended cluster may only be implemented on an endpoint that also implements both its associated BACnet Regular cluster and its associated Basic cluster.

The clusters specified herein are for use typically in Commercial Building applications, but may be used in any application domain.

9.4.2 Analog Input (BACnet Regular)

The Analog Input (BACnet Regular) cluster provides an interface for accessing a number of commonly used BACnet based attributes of an analog measurement. It is used principally for interworking with BACnet systems.

9.4.2.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.2.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BAI	Type 2 (server to client)

9.4.2.3 Cluster Identifiers

Identifier	Name
0x0602	Analog Input (BACnet Regular)

9.4.2.4 Server

9.4.2.4.1 Dependencies

Any endpoint that supports this cluster must support the Analog Input (Basic) cluster.

9.4.2.4.2 Attributes

The attributes of this cluster are detailed in Table 9-6.

Table 9-6. Attributes of the Analog Input (BACnet Regular) Server

Identifier	Name	Type	Range	Acc	Default	M/O
0x0016	*COVIncrement*	single	-	R*W	0	O
0x001F	*DeviceType*	string	-	R	Null string	O
0x004B	*ObjectIdentifier*	bacOID	0x00000000-0xffffffff	R	-	M
0x004D	*ObjectName*	string	-	R	Null string	M
0x004F	*ObjectType*	enum16	-	R	-	M
0x0076	*UpdateInterval*	uint8	-	R*W	0	O
0x00A8	*ProfileName*	string	-	R*W	Null string	O

For an explanation of the attributes, see section 9.4.20.

9.4.2.4.3 Commands

No cluster specific commands are received or generated.

9.4.2.4.4 Attribute Reporting

No attribute reporting is mandated for this cluster.

9.4.2.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.3 Analog Input (BACnet Extended)

The Analog Input (BACnet Extended) cluster provides an interface for accessing a number of BACnet based attributes of an analog measurement. It is used principally for interworking with BACnet systems.

9.4.3.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.3.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AIBE	Type 2 (server to client)

9.4.3.3 Cluster Identifiers

Identifier	Name
0x0603	Analog Input (BACnet Extended)

9.4.3.4 Server

9.4.3.4.1 Dependencies

Any endpoint that supports this cluster must support the Analog Input (Basic) cluster and the Analog Input (BACnet Regular) cluster.

9.4.3.4.2 Attributes

The attributes of this cluster are detailed in Table 9-7.

Table 9-7. Attributes of the Analog Input (BACnet Extended) Server

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*AckedTransitions*	map8	-	R*W	0	M
0x0011	*NotificationClass*	uint16	0x0000 - 0xffff	R*W	0	M
0x0019	*Deadband*	single	-	R*W	0	M
0x0023	*EventEnable*	map8	-	R*W	0	M
0x0024	*EventState*	enum8	-	R	0	O
0x002D	*HighLimit*	single	-	R*W	0	M
0x0034	*LimitEnable*	map8	0x00 - 0x11	R*W	0x00	M
0x003B	*LowLimit*	single	-	R*W	0	M
0x0048	*NotifyType*	enum8	-	R*W	0	M
0x0071	*TimeDelay*	uint8	-	R*W	0	M
0x0082	*EventTimeStamps*	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

For an explanation of the attributes, see section 9.4.20 and 9.4.21.

9.4.3.4.3 Commands

No cluster specific commands are received or generated.

9.4.3.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.4 Analog Output (BACnet Regular)

The Analog Output (BACnet Regular) cluster provides an interface for accessing a number of commonly used BACnet based attributes of an analog output. It is used principally for interworking with BACnet systems.

9.4.4.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.4.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AOB	Type 2 (server to client)

9.4.4.3 Cluster Identifiers

Identifier	Name
0x0604	Analog Output (BACnet Regular)

9.4.4.4 Server

9.4.4.4.1 Dependencies

Any endpoint that supports this cluster shall also support the Analog Output (Basic) cluster, and this cluster shall support the *PriorityArray* and *RelinquishDefault* attributes.

9.4.4.4.2 Attributes

The attributes of this cluster are detailed in Table 9-8.

Table 9-8. Attributes of the Analog Output (BACnet Regular) Server

Id	Name	Type	Range	Acc	Default	M/O
0x0016	*COVIncrement*	single	-	R*W	0	O
0x001F	*DeviceType*	string	-	R	Null string	O
0x004B	*ObjectIdentifier*	bacOID	0x00000000 - 0xffffffff	R	-	M
0x004D	*ObjectName*	string	-	R	Null string	M
0x004F	*ObjectType*	enum16	-	R	-	M
0x00A8	*ProfileName*	string	-	R*W	Null string	O

For an explanation of the attributes, see section 9.4.20.

9.4.4.4.3 Commands

No cluster specific commands are received or generated.

9.4.4.4.4 Attribute Reporting

No attribute reporting is mandated for this cluster.

9.4.4.5 Client

The client has no dependencies, no specific attributes, and receives or generates no cluster specific commands.

9.4.5 Analog Output (BACnet Extended)

The Analog Output (BACnet Extended) cluster provides an interface for accessing a number of BACnet based attributes of an analog output. It is used principally for interworking with BACnet systems.

9.4.5.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.5.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AOBE	Type 2 (server to client)

9.4.5.3 Cluster Identifiers

Identifier	Name
0x0605	Analog Output (BACnet Extended)

9.4.5.4 Server

9.4.5.4.1 Dependencies

Any endpoint that supports this cluster must support the Analog Output (Basic) cluster and the Analog Output (BACnet Regular) cluster.

9.4.5.4.2 Attributes

The attributes of this cluster are detailed in Table 9-9.

Table 9-9. Attributes of the Analog Output (BACnet Extended) Server

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*AckedTransitions*	map8	-	R*W	0	M
0x0011	*NotificationClass*	uint16	0x0000 - 0xffff	R*W	0	M
0x0019	*Deadband*	single	-	R*W	0	M
0x0023	*EventEnable*	map8	-	R*W	0	M
0x0024	*EventState*	enum8	-	R	0	O
0x002D	*HighLimit*	single	-	R*W	0	M
0x0034	*LimitEnable*	map8	0x00 - 0x11	R*W	0x00	M
0x003B	*LowLimit*	single	-	R*W	0	M
0x0048	*NotifyType*	enum8	-	R*W	0	M
0x0071	*TimeDelay*	uint8	-	R*W	0	M
0x0082	*EventTimeStamps*	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

9.4.5.4.3 Commands

No cluster specific commands are received or generated.

9.4.5.4.4 Attribute Reporting

No attribute reporting is mandated for this cluster.

9.4.5.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.6 Analog Value (BACnet Regular)

The Analog Value (BACnet Regular) cluster provides an interface for accessing commonly used BACnet based characteristics of an analog value, typically used as a control system parameter. It is principally used for interworking with BACnet systems.

9.4.6.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.6.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AVB	Type 2 (server to client)

9.4.6.3 Cluster Identifiers

Identifier	Name
0x0606	Analog Value (BACnet Regular)

9.4.6.4 Server

9.4.6.4.1 Dependencies

Any endpoint that supports this cluster must support the Analog Value (Basic) cluster.

9.4.6.4.2 Attributes

The attributes of this cluster are detailed in Table 9-10.

Table 9-10. Attributes of the Analog Value (BACnet Regular) Server

Id	Name	Type	Range	Acc	Default	M/O
0x0016	*COVIncrement*	single	-	R*W	0	O
0x004B	*ObjectIdentifier*	bacOID	0x00000000 - 0xffffffff	R	-	M
0x004D	*ObjectName*	string	-	R	Null string	M
0x004F	*ObjectType*	enum16	-	R	-	M
0x00A8	*ProfileName*	string	-	R*W	Null string	O

For an explanation of the attributes, see section 9.4.20.

9.4.6.4.3 Commands

No cluster specific commands are received or generated.

9.4.6.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.7 Analog Value (BACnet Extended)

The Analog Value (BACnet Extended) cluster provides an interface for accessing BACnet based characteristics of an analog value, typically used as a control system parameter. It is principally used for interworking with BACnet systems.

9.4.7.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.7.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AVBE	Type 2 (server to client)

9.4.7.3 Cluster Identifiers

Identifier	Name
0x0607	Analog Value (BACnet Extended)

9.4.7.4 Server

9.4.7.4.1 Dependencies

Any endpoint that supports this cluster must support the Analog Value (Basic) cluster and the Analog Value (BACnet Regular) cluster.

9.4.7.4.2 Attributes

The attributes of this cluster are detailed in Table 9-11.

Table 9-11. Attributes of the Analog Value (BACnet Extended) Server

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*AckedTransitions*	map8	-	R*W	0	M
0x0011	*NotificationClass*	uint16	0x0000 - 0xffff	R*W	0	M
0x0019	*Deadband*	single	-	R*W	0	M
0x0023	*EventEnable*	map8	-	R*W	0	M
0x0024	*EventState*	enum8	-	R	0	O
0x002D	*HighLimit*	single	-	R*W	0	M
0x0034	*LimitEnable*	map8	0x00 - 0x11	R*W	0x00	M
0x003B	*LowLimit*	single	-	R*W	0	M
0x0048	*NotifyType*	enum8	-	R*W	0	M
0x0071	*TimeDelay*	uint8	-	R*W	0	M
0x0082	*EventTimeStamps*	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

For an explanation of the attributes, see sections9.4.20 and 9.4.21.

9.4.7.4.3 Commands

No cluster specific commands are received or generated.

9.4.7.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.8 Binary Input (BACnet Regular)

The Binary Input (BACnet Regular) cluster provides an interface for accessing a number of commonly used BACnet based attributes of a binary measurement. It is used principally for interworking with BACnet systems.

9.4.8.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.8.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BIB	Type 2 (server to client)

9.4.8.3 Cluster Identifiers

Identifier	Name
0x0608	Binary Input (BACnet Regular)

9.4.8.4 Server

9.4.8.4.1 Dependencies

Any endpoint that supports this cluster must support the Binary Input (Basic) cluster.

9.4.8.4.2 Attributes

The attributes of this cluster are detailed in Table 9-12.

Table 9-12. Attributes of the Binary Input (BACnet Regular) Server

Id	Name	Type	Range	Acc	Default	MO
0x000F	*ChangeOfStateCount*	uint32	-	R*W	0xffffffff	O
0x0010	*ChangeOfStateTime*	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O

Id	Name	Type	Range	Acc	Default	MO
0x001F	*DeviceType*	string	-	R	Null string	O
0x0021	*ElapsedActiveTime*	uint32	-	R*W	0xffffffff	O
0x004B	*ObjectIdentifier*	bacOID	0x00000000 - 0xffffffff	R	-	M
0x004D	*ObjectName*	string	-	R	Null string	M
0x004F	*ObjectType*	enum16	-	R	-	M
0x0072	*TimeOfATReset*	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x0073	*TimeOfSCReset*	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x00A8	*ProfileName*	string	-	R*W	Null string	O

For an explanation of the attributes, see section 9.4.20.

9.4.8.4.3 Commands

No cluster specific commands are received or generated.

9.4.8.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.9 Binary Input (BACnet Extended)

The Binary Input (BACnet Extended) cluster provides an interface for accessing a number of BACnet based attributes of a binary measurement. It is used principally for interworking with BACnet systems.

9.4.9.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.9.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BIBE	Type 2 (server to client)

9.4.9.3 Cluster Identifiers

Identifier	Name
0x0609	Binary Input (BACnet Extended)

9.4.9.4 Server

9.4.9.4.1 Dependencies

Any endpoint that supports this cluster must support the Binary Input (Basic) cluster and the Binary Input (BACnet Regular) cluster.

9.4.9.4.2 Attributes

The attributes of this cluster are detailed in Table 9-13.

Table 9-13. Attributes of the Binary Input (BACnet Extended) Server

Id	Name	Type	Range	Access	Def	M/O
0x0000	*AckedTransitions*	map8	-	R*W	0	M
0x0006	*AlarmValue*	bool	0 - 1	R*W	-	M
0x0011	*NotificationClass*	uint16	0x0000 - 0xffff	R*W	0	M
0x0023	*EventEnable*	map8	-	R*W	0	M
0x0024	*EventState*	enum8	-	R	0	O
0x0048	*NotifyType*	enum8	-	R*W	0	M
0x0071	*TimeDelay*	uint8	-	R*W	0	M
0x0082	*EventTimeStamps*	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

9.4.9.4.3 Commands

No cluster specific commands are received or generated.

9.4.9.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.10 Binary Output (BACnet Regular)

The Analog Output (BACnet Regular) cluster provides an interface for accessing a number of commonly used BACnet based attributes of a binary output. It is used principally for interworking with BACnet systems.

9.4.10.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.10.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BOB	Type 2 (server to client)

9.4.10.3 Cluster Identifiers

Identifier	Name
0x060a	Binary Output (BACnet Regular)

9.4.10.4 Server

9.4.10.4.1 Dependencies

Any endpoint that supports this cluster shall also support the Binary Output (Basic) cluster, and this cluster shall support the PriorityArray and RelinquishDefault attributes.

9.4.10.4.2 Attributes

The attributes of this cluster are detailed in Table 9-14.

Table 9-14. Attributes of the Binary Output (BACnet Regular) Server

Id	Name	Type	Range	Acc	Default	MO
0x000F	*ChangeOfStateCount*	uint32	-	R*W	0xffffffff	O
0x0010	*ChangeOfStateTime*	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x001F	*DeviceType*	string	-	R	Null string	O
0x0021	*ElapsedActiveTime*	uint32	-	R*W	0xffffffff	O
0x0028	*FeedBackValue*	enum8	0 - 1	R*W	0	O
0x004B	*ObjectIdentifier*	bacOID	0x00000000 - 0xffffffff	R	-	M

Id	Name	Type	Range	Acc	Default	MO
0x004D	*ObjectName*	string	-	R	Null string	M
0x004F	*ObjectType*	enum16	-	R	-	M
0x0072	*TimeOfATReset*	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x0073	*TimeOfSCReset*	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x00A8	*ProfileName*	string	-	R*W	Null string	O

For an explanation of the attributes, see section 9.4.20.

9.4.10.4.3 Commands

No cluster specific commands are received or generated.

9.4.10.4.4 Attribute Reporting

No attribute reporting is mandated for this cluster.

9.4.10.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.11 Binary Output (BACnet Extended)

The Binary Output (BACnet Extended) cluster provides an interface for accessing a number of BACnet based attributes of a binary output. It is used principally for interworking with BACnet systems.

9.4.11.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.11.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BOBE	Type 2 (server to client)

9.4.11.3 Cluster Identifiers

Identifier	Name
0x060b	Binary Output (BACnet Extended)

9.4.11.4 Server

9.4.11.4.1 Dependencies

Any endpoint that supports this cluster must support the Binary Output (Basic) cluster and the Binary Output (BACnet Regular) cluster.

9.4.11.4.2 Attributes

The attributes of this cluster are detailed in Table 9-15.

Table 9-15. Attributes of the Binary Output (BACnet Extended) Server

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*AckedTransitions*	map8	-	R*W	0	M
0x0011	*NotificationClass*	uint16	0x0000 - 0xffff	R*W	0	M
0x0023	*EventEnable*	map8	-	R*W	0	M
0x0024	*EventState*	enum8	-	R	0	O
0x0048	*NotifyType*	enum8	-	R*W	0	M
0x0071	*TimeDelay*	uint8	-	R*W	0	M
0x0082	*EventTimeStamps*	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

9.4.11.4.3 Commands

No cluster specific commands are received or generated.

9.4.11.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.12 Binary Value (BACnet Regular)

The Binary Value (BACnet Regular) cluster provides an interface for accessing commonly used BACnet based characteristics of a binary value, typically used as a control system parameter. It is principally used for interworking with BACnet systems.

9.4.12.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.12.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BVB	Type 2 (server to client)

9.4.12.3 Cluster Identifiers

Identifier	Name
0x060c	Binary Value (BACnet Regular)

9.4.12.4 Server

9.4.12.4.1 Dependencies

Any endpoint that supports this cluster must support the Binary Value (Basic) cluster.

9.4.12.4.2 Attributes

The attributes of this cluster are detailed in Table 9-16.

Table 9-16. Attributes of the Binary Value (BACnet Regular) Server

Id	Name	Type	Range	Acc	Default	M/O
0x000F	*ChangeOfStateCount*	uint32	-	R*W	0xffffffff	O
0x0010	*ChangeOfStateTime*	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x0021	*ElapsedActiveTime*	uint32	-	R*W	0xffffffff	O
0x004B	*ObjectIdentifier*	bacOID	0-0xffffffff	R	-	M
0x004D	*ObjectName*	string	-	R	Null string	M
0x004F	*ObjectType*	enum16	-	R	-	M
0x0072	*TimeOfATReset*	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x0073	*TimeOfSCReset*	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x00A8	*ProfileName*	string	-	R*W	Null string	O

For an explanation of the attributes, see section 9.4.20.

9.4.12.4.3 Commands

No cluster specific commands are received or generated.

9.4.12.4.4 Attribute Reporting

No attribute reporting is mandated for this cluster.

9.4.12.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.13 Binary Value (BACnet Extended)

The Binary Value (BACnet Extended) cluster provides an interface for accessing BACnet based characteristics of a binary value, typically used as a control system parameter. It is principally used for interworking with BACnet systems.

9.4.13.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.13.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BVBE	Type 2 (server to client)

9.4.13.3 Cluster Identifiers

Identifier	Name
0x060d	Binary Value (BACnet Extended)

9.4.13.4 Server

9.4.13.4.1 Dependencies

Any endpoint that supports this cluster must support the Binary Value (Basic) cluster and the Binary Value (BACnet Regular) cluster.

9.4.13.4.2 Attributes

The attributes of this cluster are detailed in Table 9-17.

Table 9-17. Attributes of the Binary Value (BACnet Extended) Server

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*AckedTransitions*	map8	-	R*W	0	M
0x0006	*AlarmValue*	bool	0 - 1	R*W	-	M
0x0011	*NotificationClass*	uint16	0x0000 - 0xffff	R*W	0	M

Id	Name	Type	Range	Acc	Def	M/O
0x0023	*EventEnable*	map8	-	R*W	0	M
0x0024	*EventState*	enum8	-	R	0	O
0x0048	*NotifyType*	enum8	-	R*W	0	M
0x0071	*TimeDelay*	uint8	-	R*W	0	M
0x0082	*EventTimeStamps*	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

9.4.13.4.3 Commands

No cluster specific commands are received or generated.

9.4.13.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.14 Multistate Input (BACnet Regular)

The Multistate Input (BACnet Regular) cluster provides an interface for accessing a number of commonly used BACnet based attributes of a multistate measurement. It is used principally for interworking with BACnet systems.

9.4.14.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.14.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MIB	Type 2 (server to client)

9.4.14.3 Cluster Identifiers

Identifier	Name
0x060e	Multistate Input (BACnet Regular)

9.4.14.4 Server

9.4.14.4.1 Dependencies

Any endpoint that supports this cluster must support the Multistate Input (Basic) cluster.

9.4.14.4.2 Attributes

The attributes of this cluster are detailed in Table 9-18.

Table 9-18. Attributes of the Multistate Input (BACnet Regular) Server

Id	Name	Type	Range	Acc	Default	M/O
0x001F	*DeviceType*	string	-	R	Null string	O
0x004B	*ObjectIdentifier*	bacOID	0-0xffffffff	R	-	M
0x004D	*ObjectName*	string	-	R	Null string	M
0x004F	*ObjectType*	enum16	-	R	-	M
0x00A8	*ProfileName*	string	-	R*W	Null string	O

For an explanation of the attributes, see section 9.4.20.

9.4.14.4.3 Commands

No cluster specific commands are received or generated.

9.4.14.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.15 Multistate Input (BACnet Extended)

The Multistate Input (BACnet Extended) cluster provides an interface for accessing a number of BACnet based attributes of a multistate measurement. It is used principally for interworking with BACnet systems.

9.4.15.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.15.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MIBE	Type 2 (server to client)

9.4.15.3 Cluster Identifiers

Identifier	Name

0x060f	Multistate Input (BACnet Extended)

9.4.15.4 Server

9.4.15.4.1 Dependencies

Any endpoint that supports this cluster must support the Multistate Input (Basic) cluster and the Multistate Input (BACnet Regular) cluster.

9.4.15.4.2 Attributes

The attributes of this cluster are detailed in Table 9-19.

Table 9-19. Attributes of Multistate Input (BACnet Extended) Server

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*AckedTransitions*	map8	-	R*W	0	M
0x0006	*AlarmValues*	Set of uint16	0 - 0xffff	R*W	-	M
0x0011	*NotificationClass*	uint16	0x0000 - 0xffff	R*W	0	M
0x0023	*EventEnable*	map8	-	R*W	0	M
0x0024	*EventState*	enum8	-	R	0	O
0x0025	*FaultValues*	Set of uint16	0 - 0xffff	R*W	0	M
0x0048	*NotifyType*	enum8	-	R*W	0	M
0x0071	*TimeDelay*	uint8	-	R*W	0	M
0x0082	*EventTimeStamps*	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

9.4.15.4.3 Commands

No cluster specific commands are received or generated.

9.4.15.4.4 Attribute Reporting

No attribute reporting is mandated for this cluster.

9.4.15.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.16 Multistate Output (BACnet Regular)

The Multistate Output (BACnet Regular) cluster provides an interface for accessing a number of commonly used BACnet based attributes of a multistate output. It is used principally for interworking with BACnet systems.

9.4.16.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.16.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MOB	Type 2 (server to client)

9.4.16.3 Cluster Identifiers

Identifier	Name
0x0610	Multistate Output (BACnet Regular)

9.4.16.4 Server

9.4.16.4.1 Dependencies

Any endpoint that supports this cluster shall also support the Multistate Output (Basic) cluster, and this cluster shall support the PriorityArray and RelinquishDefault attributes.

9.4.16.4.2 Attributes

The attributes of this cluster are detailed in Table 9-20.

Table 9-20. Attributes of Multistate Output (BACnet Regular) Server

Id	Name	Type	Range	Access	Default	M/O
0x001F	*DeviceType*	string	-	R	Null string	O
0x0028	*FeedBackValue*	enum8	0 - 1	R*W	0	O
0x004B	*ObjectIdentifier*	bacOID	0x00000000 - 0xffffffff	R	-	M
0x004D	*ObjectName*	string	-	R	Null string	M
0x004F	*ObjectType*	enum16	-	R	-	M
0x00A8	*ProfileName*	string	-	R*W	Null string	O

For an explanation of the attributes, see section 9.4.20.

9.4.16.4.3 Commands

No cluster specific commands are received or generated.

9.4.16.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.17 Multistate Output (BACnet Extended)

The Multistate Output (BACnet Extended) cluster provides an interface for accessing a number of BACnet based attributes of a multistate output. It is used principally for interworking with BACnet systems.

9.4.17.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.17.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MOBE	Type 2 (server to client)

9.4.17.3 Cluster Identifiers

Identifier	Name
0x0611	Multistate Output (BACnet Extended)

9.4.17.4 Server

9.4.17.4.1 Dependencies

Any endpoint that supports this cluster must support the Multistate Output (Basic) cluster and the Multistate Output (BACnet Regular) cluster.

9.4.17.4.2 Attributes

The attributes of this cluster are detailed in Table 9-21.

Table 9-21. Attributes of Multistate Output (BACnet Extended) Server

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*AckedTransitions*	map8	-	R*W	0	M
0x0011	*NotificationClass*	uint16	0x0000 - 0xffff	R*W	0	M
0x0023	*EventEnable*	map8	-	R*W	0	M
0x0024	*EventState*	enum8	-	R	0	O
0x0048	*NotifyType*	enum8	-	R*W	0	M
0x0071	*TimeDelay*	uint8	-	R*W	0	M

Id	Name	Type	Range	Acc	Def	M/O
0x0082	*EventTimeStamps*	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

9.4.17.4.3 Commands

No cluster specific commands are received or generated.

9.4.17.4.4 Attribute Reporting

No attribute reporting is mandated for this cluster.

9.4.17.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.18 Multistate Value (BACnet Regular)

The Multistate Value (BACnet Regular) cluster provides an interface for accessing commonly used BACnet based characteristics of a multistate value, typically used as a control system parameter. It is principally used for interworking with BACnet systems.

9.4.18.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.18.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MVB	Type 2 (server to client)

9.4.18.3 Cluster Identifiers

Identifier	Name
0x0612	Multistate Value (BACnet Regular)

9.4.18.4 Server

9.4.18.4.1 Dependencies

Any endpoint that supports this cluster must support the Multistate Value (Basic) cluster.

9.4.18.4.2 Attributes

The attributes of this cluster are detailed in Table 9-22.

Table 9-22. Attributes of Multistate Value (BACnet Regular) Server

Id	Name	Type	Range	Acc	Default	M/O
0x004B	*ObjectIdentifier*	bacOID	0 -0xffffffff	R	-	M
0x004D	*ObjectName*	string	-	R	Null string	M
0x004F	*ObjectType*	enum16	-	R	-	M
0x00A8	*ProfileName*	string	-	R*W	Null string	O

For an explanation of the attributes, see section 9.4.20.

9.4.18.4.3 Commands

No cluster specific commands are received or generated.

9.4.18.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.19 Multistate Value (BACnet Extended)

The Multistate Value (BACnet Extended) cluster provides an interface for accessing BACnet based characteristics of a multistate value, typically used as a control system parameter. It is principally used for interworking with BACnet systems.

9.4.19.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.4.19.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MVBE	Type 2 (server to client)

9.4.19.3 Cluster Identifiers

Identifier	Name
0x0613	Multistate Value (BACnet Extended)

9.4.19.4 Server

9.4.19.4.1 Dependencies

Any endpoint that supports this cluster must support the Multistate Value (Basic) cluster and the Multistate Value (BACnet Regular) cluster.

9.4.19.4.2 Attributes

The attributes of this cluster are detailed in Table 9-23.

Table 9-23. Attributes of Multistate Value (BACnet Extended) Server

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*AckedTransitions*	map8	-	R*W	0	M
0x0006	*AlarmValues*	set of uint16	0 - 0xffff	R*W	-	M
0x0011	*NotificationClass*	uint16	0x0000 - xffff	R*W	0	M
0x0023	*EventEnable*	map8	-	R*W	0	M
0x0024	*EventState*	enum8	-	R	0	O
0x0025	*FaultValues*	set of uint16	0 - 0xffff	R*W	0	M
0x0048	*NotifyType*	enum8	-	R*W	0	M
0x0071	*TimeDelay*	uint8	-	R*W	0	M
0x0082	*EventTimeStamps*	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

9.4.19.4.3 Commands

No cluster specific commands are received or generated.

9.4.19.4.4 Attribute Reporting

No attribute reporting is mandated for this cluster.

9.4.19.5 Client

The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

9.4.20 Attributes of BACnet Regular Clusters

The attributes of BACnet Regular and BACnet Extended clusters are specifically intended for interworking with BACnet systems (via a BACnet gateway). They are based on BACnet properties with the same names. See the BACnet Reference Manual [A1] for detailed descriptions of these properties.

References to reports in this section refer to BACnet intrinsic reporting. Note that attribute reporting may be used to send reports as well.

9.4.20.1 ObjectIdentifier Attribute

This attribute, of type BACnet OID, is a numeric code that is used to identify the object. It shall be unique within the BACnet Device that maintains it.

9.4.20.2 *ObjectName* Attribute

This attribute, of type Character String, shall represent a name for the object that is unique within the BACnet Device that maintains it. The minimum length of the string shall be one character. The set of characters used in the *ObjectName* shall be restricted to printable characters.

9.4.20.3 *ObjectType* Attribute

This attribute, of type enumeration, is set to the ID of the corresponding BACnet object type from which the cluster was derived.

9.4.20.4 COVIncrement Attribute

This attribute, of type single, specifies the minimum change in *PresentValue* that will cause a value change report to be initiated to bound report recipient clients. This value is the same as the Reportable Change value for the *PresentValue* attribute.

9.4.20.5 *DeviceType* Attribute

This attribute, of type Character String, is a text description of the physical device connected to the input, output or value.

9.4.20.6 UpdateInterval Attribute

This attribute indicates the maximum period of time between updates to the *PresentValue* of an Analog Input cluster, in hundredths of a second, when the input is not overridden and not out-of-service.

9.4.20.7 ChangeOfStateCount Attribute

This attribute, of type Unsigned 32-bit integer, represents the number of times that the *PresentValue* attribute of a Binary Input, Output or Value cluster has changed state (from 0 to 1, or from 1 to 0) since the *ChangeOfStateCount* attribute was most recently set to a zero value. The *ChangeOfStateCount* attribute shall have a range of 0-65535 or greater.

When *OutOfService* is FALSE, a change to the *Polarity* attribute shall alter *PresentValue* and thus be considered a change of state. When *OutOfService* is TRUE, changes to *Polarity* shall not cause changes of state. If one of the optional attributes *ChangeOfStateTime*, *ChangeOfStateCount*, or *TimeOfStateCountReset* is present, then all of these attributes shall be present.

9.4.20.8 ChangeOfStateTime Attribute

This attribute, of type Structure (Date, Time of Day), represents the most recent date and time at which the *PresentValue* attribute of a Binary Input, Output or Value cluster changed state (from 0 to 1, or from 1 to 0)

When *OutOfService* is FALSE, a change to the *Polarity* attribute shall alter *PresentValue* and thus be considered a change of state. When *OutOfService* is TRUE, changes to *Polarity* shall not cause changes of state. If one of the optional attributes *ChangeOfStateTime*, *ChangeOfStateCount*, or *TimeOfSCReset* is present, then all of these attributes shall be present.

9.4.20.9 ElapsedActiveTime Attribute

This attribute, of type Unsigned 32-bit integer, represents the accumulated number of seconds that the *PresentValue* attribute of a Binary Input, Output or Value cluster has had the value ACTIVE (1) since the *ElapsedActiveTime* attribute was most recently set to a zero value. If one of the optional properties *ElapsedActiveTime* or *TimeOfATReset* is present, then both of these attributes shall be present.

9.4.20.10 TimeOfATReset Attribute

This attribute, of type Structure (Date, Time of Day), represents the date and time at which the *ElapsedActiveTime* attribute of a Binary Input, Output or Value cluster was most recently set to a zero value. If one of the optional properties *ElapsedActiveTime* or *TimeOfATReset* is present, then both of these attributes shall be present.

9.4.20.11 TimeOfSCReset Attribute

This attribute, of type Structure (Date, Time of Day), represents the date and time at which the *ChangeOfStateCount* attribute of a Binary Input, Output or Value cluster was most recently set to a zero value. If one of the optional properties *ChangeOfStateTime*, *ChangeOfStateCount*, or *TimeOfSCReset* is present, then all of these attributes shall be present.

9.4.20.12 FeedbackValue Attribute

This property, of type enumeration, indicates a feedback value from which *PresentValue* must differ before an OFFNORMAL event is generated, and to which *PresentValue* must return before a TONORMAL event is generated. The manner by which the *FeedbackValue* is determined shall be a local matter.

9.4.20.13 ProfileName Attribute

This attribute, of type Character string, is the name of a BACnet object profile to which its associated cluster conforms. A profile defines a set of additional attributes, behavior, and/or requirements for the cluster beyond those specified here.

To ensure uniqueness, a profile name must begin with a vendor identifier code (see Clause 23 of [A1]) in base-10 integer format, followed by a dash. All subsequent characters are administered by the organization registered with that vendor identifier code. The vendor identifier code that prefixes the profile name shall indicate the organization that publishes and maintains the profile document named by the remainder of the profile name. This vendor identifier need not have any relationship to the vendor identifier of the device within which the object resides.

9.4.21 Attributes of BACnet Extended Clusters

The attributes of BACnet Extended clusters are specifically intended for interworking with BACnet systems (via a BACnet gateway). They are based on BACnet properties with the same names. See the BACnet Reference Manual [A1] for detailed descriptions of these properties.

References to events and alarms in this section refer to BACnet intrinsic reporting. Note that attribute reporting may be used to send reports as well.

9.4.21.1 AckedTransitions Attribute

This attribute, of type bitmap, holds three one-bit flags (b0, b1, b2) that respectively indicate the receipt of acknowledgments for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events

9.4.21.2 *AlarmValue* Attribute

This attribute, of type Boolean, specifies the value that the *PresentValue* attribute must have before a TO-OFFNORMAL event is generated.

9.4.21.3 AlarmValues Attribute

This attribute, of type Set of uint16, specifies any values that the *PresentValue* attribute must equal before a TO-OFFNORMAL event is generated.

9.4.21.4 FaultValues Attribute

This attribute, of type Set of uint16, specifies any values that the *PresentValue* attribute must equal before a TO-FAULT event is generated.

9.4.21.5 NotificationClass Attribute

This attribute, of type uint16, specifies the notification class to be used when handling and generating event notifications for this object (over a BACnet gateway).

9.4.21.6 *Deadband* Attribute

This attribute, of type single, specifies a range (from *LowLimit* + *Deadband* to *HighLimit* - *Deadband*) which the *PresentValue* must return within for a TO-NORMAL event to be generated.

9.4.21.7 EventEnable Attribute

This attribute, of type bitmap, holds three one-bit flags (b0, b1, b2) that respectively enable (1) and disable (0) reporting of TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events.

9.4.21.8 *EventState* Attribute

The *EventState* attribute, of type 8-bit enumeration, is included in order to provide a way to determine if this object has an active event state associated with it. The allowed values are:

- NORMAL (0)
- FAULT (1)
- OFFNORMAL (2)
- HIGH-LIMIT (3)
- LOW-LIMIT (4)

9.4.21.9 *HighLimit* Attribute

This attribute, of type single, specifies a limit that *PresentValue* must exceed before an OFF-NORMAL (HIGH-LIMIT) event is generated.

9.4.21.10 LimitEnable Attribute

This attribute, of type map8, holds two one-bit flags. The flag in bit position 0 enables reporting of low limit off-normal and return-to-normal events if it has the value 1, and disables reporting of these events if it has the value 0. The flag in bit position 1 enables reporting of high limit off-normal and return-to-normal events if it has the value 1, and disables reporting of these events if it has the value 0.

9.4.21.11 *LowLimit* Attribute

This attribute, of type single, shall specify a limit that *PresentValue* must fall below before an OFF-NORMAL (LOW-LIMIT) event is generated.

9.4.21.12 *NotifyType* Attribute

This attribute, of type enumeration, indicates whether the notifications generated by the cluster should be Events (0) or Alarms (1).

9.4.21.13 *TimeDelay* Attribute

This attribute, of type Unsigned 8-bit integer, specifies the minimum period of time in seconds that *PresentValue* must remain outside the band defined by the *HighLimit* and *LowLimit* attributes before a TO-OFFNORMAL event is generated, or within the band (from *LowLimit* + *Deadband* to *HighLimit* - *Deadband*) before a TO-NORMAL event is generated.

9.4.21.14 EventTimeStamps Attribute

This optional read-only attribute is of type Array[3]. The three elements each have a type which is one of:

- 16-bit unsigned integer - a sequence number
- Time of day
- Structure of (date, time of day)

The elements of the array hold the times (or sequence numbers) of the last event notifications for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events, respectively. The type of the elements is discovered by reading the attribute.

9.5 ISO 7818 Protocol Tunnel

9.5.1 Scope and Purpose

This section specifies a single cluster, the ISO7816 Tunnel cluster, which provides commands and attributes for mobile office solutions.

This cluster is to provide a standardized interface to enable a scenario of authorization management on mobile office devices (e.g., access to PC resources)

9.5.2 Definitions

The definitions used in the ISO 7816 Protocol Tunnel are shown in Table 9-24.

Table 9-24. Definitions Used in ISO 7816 Protocol Tunnel Description

Term	Definition

Target Device	A Computer System on which User has to perform authentication in order to access to information services
User Token	A device used by a Target Device to authenticate and authorize User
Virtual SmartCard	A SmartCard that is a node on the network

9.5.3 General Description

The cluster specified in this document is typically used for telecom applications, but may be used in any other application domains.

9.5.4 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides attributes and commands to tunnel ISO7816 APDUs, enabling solution such as Mobile Office, i.e., a mechanism to authenticate and authorize Users on shared Computer System (said Target Device) by means of a Virtual Smartcard (generically said User Token).

A Target Device, enabled by the server side of this cluster, and a User Token (supporting a client side of this cluster) can establish a connection and exchange information by means of ISO7816 APDU messages over network.

9.5.4.1 Revision History

Rev	**Description**
1	mandatory global *ClusterRevision* attribute added

9.5.4.2 Classification

Hierarchy	**Role**	**PICS Code**	**Primary Transaction**
Base	Application	T7816	Type 1 (client to server)

9.5.4.3 Cluster Identifiers

Identifier	**Name**
0x0615	ISO 7818 Protocol Tunnel

9.5.5 Server

9.5.5.1 Dependencies

Since ISO7816 protocol may use APDU frames larger than typical payload, stack fragmentation or Partition cluster shall be supported by the devices supporting this cluster.

9.5.5.2 Attributes

The ISO7816 Tunnel cluster contains the attribute shown in Table 9-25.

Table 9-25. Attributes for the ISO7816 Tunnel Cluster

Id	Name	Type	Range	Access	Default	M/O
0x0001	*Status*	uint8	0x00-0x01	R	0x00	M

9.5.5.2.1 *Status* Attribute

The *Status* attribute specifies the Server internal state.

Values and usage of this attribute are application dependent, e.g., server busy (client connected). Server supports only one client connection at a time.

The *Status* values are shown in Table 9-26.

Table 9-26. Status Values

Meaning	Values
0x00	FREE
0x01	BUSY

9.5.5.3 Commands Received

The cluster specific commands received by the ISO7816 Tunnel server cluster are listed in Table 9-27.

Table 9-27. Received Command IDs for the ISO7816 Tunnel Cluster

Command Identifier Field Value	Description	M/O
0x00	Transfer APDU	M
0x01	Insert SmartCard	M
0x02	Extract SmartCard	M

9.5.5.3.1 Transfer APDU Command

9.5.5.3.1.1 Payload Format

The Transfer APDU command shall be formatted as illustrated in Figure 9-5.

Figure 9-5. Format of the Transfer APDU command

Bits	Variable
Data Type	octstr
Field Name	APDU

9.5.5.3.1.2 APDU Field

The APDU field is of variable length and is an ISO7816 APDU as defined in the ISO7816 standard [I2]

9.5.5.3.1.3 When Generated

This command is generated when an ISO7816 APDU has to be transferred across a tunnel.

9.5.5.3.1.4 Effect on Receipt

On receipt of this command, a device shall process the ISO7816 APDU as specified in the ISO7816 standard.

9.5.5.3.2 Insert Smart Card

9.5.5.3.2.1 Payload Format

No payload needed for Insert Smart Card command.

9.5.5.3.2.2 When Generated

This command is generated when a User Token insertion has to be sent to Server.

9.5.5.3.2.3 Effect on Receipt

On receipt of this command:

- If the *Status* attribute is equal to BUSY, the Server shall send a Default Response with status FAILURE.
- If the *Status* attribute is equal to FREE and the bit ‘Disable Default Response’ of the Frame control field of the ZCL Header is set to zero, the Server shall send respond with status SUCCESS. It also shall set its Status Attributes to BUSY and it can start to exchange APDUs with Client over ISO7816 Tunnel.

9.5.5.3.3 Extract Smart Card

9.5.5.3.3.1 Payload Format

No payload needed for Insert Smart Card command.

9.5.5.3.3.2 When Generated

This command is generated when a User Token extraction has to be sent to Server.

9.5.5.3.3.3 Effect on Receipt

On receipt of this command:

- If the *Status* attribute is equal to FREE, the Server shall send a Default Response with status FAILURE.
- If the *Status* attribute is equal to BUSY and the bit ‘Disable Default Response’ of the of the Frame control field of the ZCL Header is set to zero, the Server shall send respond with status SUCCESS. It shall also set its Status Attributes to FREE and after this, Server shall not be able to exchange APDUs with Client over ISO7816 Tunnel.

9.5.5.4 Commands Generated

The cluster specific commands generated by the ISO7816 Tunnel server cluster are listed in Table 9-28.

Table 9-28. Generated Command IDs for the ISO7816 Tunnel Cluster

Command Identifier Field Value	Description	M/O
0x00	Transfer APDU	M

9.5.5.5 Transfer APDU

9.5.5.5.1.1 Payload Format

The Transfer APDU command shall be formatted using the same command "Transfer APDU" in paragraph 9.5.5.3.1. The effect on receipt is the same as reported in 9.5.5.3.1.4.

9.5.6 Client

9.5.6.1 Dependencies

None

9.5.6.2 Attributes

The client cluster has no attributes.

9.5.6.3 Command Received

The client receives the cluster specific commands detailed in 9.5.5.4 as required by application profiles

9.5.6.4 Command Generated

The client generates the cluster specific commands detailed in 9.5.5.3 as required by application profiles.

9.6 Partition

9.6.1 Scope and Purpose

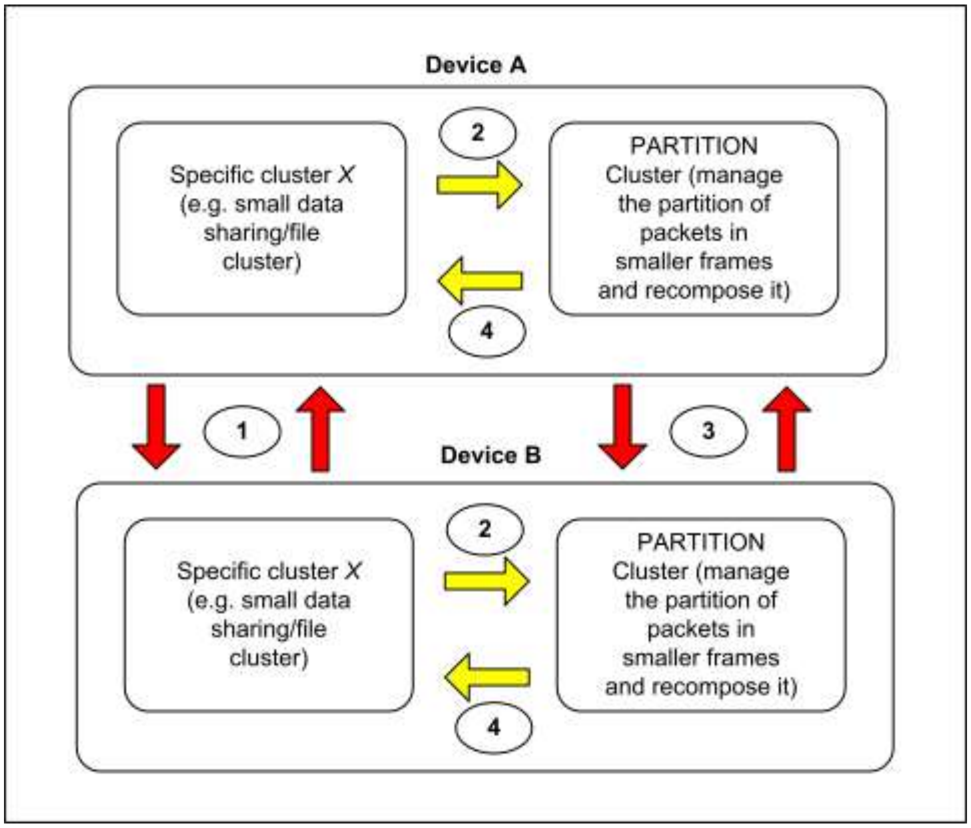
This section specifies a single cluster, the Partition cluster, which provides commands and attributes for enabling partitioning of large frame to be carried from other clusters. This cluster is designed to provide a standardized interface for the applications to manage extended size frame format, up to 100KB long. The Partition cluster can be used in different application scenarios that requires extended frame for services provided by particular clusters.

9.6.2 Introduction

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The cluster specified in this may be used in different application domains. The Partition cluster provides the attributes and commands required for enabling and managing the transmission of extended frames over a network.

Figure 9-6. Typical Usage of the Partition Cluster


Device A
Specific cluster X (e.g. small data sharing/file cluster)
2
PARTITION Cluster (manage the partition of packets in smaller frames and recompose it)
4
1
3
Device B
Specific cluster X (e.g. small data sharing/file cluster)
2
PARTITION Cluster (manage the partition of packets in smaller frames and recompose it)
4

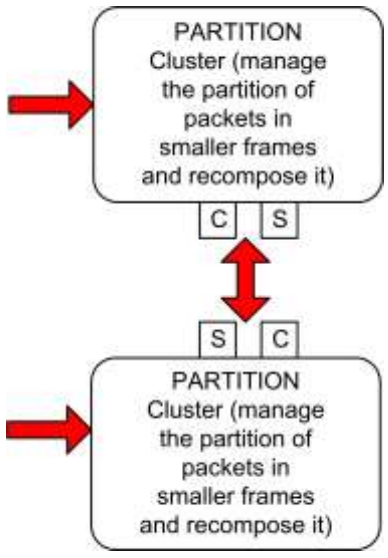
The typical usage of Partition cluster is shown in Figure 9-6 and can be represented by the following phases:

3. Cluster based Discovery (e.g., performing Match_Desc_Req) can be operated to the specific cluster X that needs to transfer information to a matching cluster (e.g., File Cluster); moreover cluster based discovery should be used in order to check the support of the Partition Cluster by a recipient device.
4. If the application entity requires transmission of large frames (e.g., an application willing to use data sharing/file cluster, generic tunnel cluster) the specific application entity shall subscribe to the Partition cluster; registration or subscription phase is described in 9.6.5.
5. The Partition clusters will perform and manage the "fragmentation" and send the rebuilt frame to the registered specific cluster.
6. The Partition Cluster will forward the recomposed packet to the specific clusters that registered to the Partitioning Cluster (e.g., Cluster X).

The application object implementing and using the Partition Cluster should have enough memory to manage the incoming frames; the Partition cluster is designed for devices like Mobile Phones or other gateways that have extended computing capabilities in comparison with typical devices.

Since the Partition cluster performs a handshake phase between the devices using Partition cluster (reading and writing the proper defined attributes) as described with more details in 9.6.5, both client and server should be used in order to guarantee a full bidirectional link in the communication (see Figure 9-7).

Figure 9-7. Client and Server in Partition Cluster


PARTITION Cluster (manage the partition of packets in smaller frames and recompose it)
C
S
S
C
PARTITION Cluster (manage the partition of packets in smaller frames and recompose it)

A simple way to enable the use of the partition cluster should be to define a specific API that would support the sending/receive functionalities through the use of *Partition Cluster.* Partition should be considered like a specific tunnel cluster: Commands exposed to the application objects (general API to be used by the application) should be the following ones:

- *TransferFrameUsingPartitionCluster* (send/receive) → the max size for the carried data is typically 25KB<*x*<100KB as from discussed requirements. This command may pass a handler to the sequence of bytes corresponding to the ZCL message of the specific cluster using the Partition Cluster. In order to operate using the Partition Cluster the application may want to manage the transmission and reception of large frames running the handshake phase described in 9.6.5.

  Rather than pushing the large frame to the application, the Partition Cluster may only inform the application that a packet has arrived (very short packet that can be fed through the stack). The application will then read the frame from the Partitioning Cluster. The detailed mechanism to perform this operation is out of scope of this specification

- *RW handshake commands*

Partition cluster related commands should be sent transparently between the application objects managing the fragmentation to guarantee the reconstruction of the received frame; these commands are described in the following sections:

- *Transfer partitioned frame* (max dimension<max size carried by the ZCL standard frame ~80B)
- *Multiple ACKs*

In the Partition Cluster attributes a list of registered clusters should be inserted in order to manage possible sharing and re-use of it by multiple clusters.

9.6.2.1 Revision History

Rev	**Description**
1	mandatory global *ClusterRevision* attribute added

9.6.2.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	PART	Type 1 (client to server)

9.6.2.3 Cluster Identifiers

Identifier	Name
0x0016	Partition

9.6.3 Server

9.6.3.1 Dependencies

None

9.6.3.2 Attributes

The attributes are used in the Partition Cluster summarized in Table 9-29.

Table 9-29. Attributes of the Partition Cluster

Id	Name	Type	Range	Acc	Default	M/O
0x0000	*MaximumIncomingTransferSize*	uint16	0x0000-0xffff	R	0x0500	M
0x0001	*MaximumOutgoingTransferSize*	uint16	0x0000-0xffff	R	0x0500	M
0x0002	*PartionedFrameSize*	uint8	0x00-0xff	RW	0x50	M
0x0003	*LargeFrameSize*	uint16	0x0000-0xffff	RW	0x0500	M
0x0004	*NumberOfACKFrame*	uint8	0x00-0xff	RW	0x64	M
0x0005	*NACKTimeout*	uint16	0x0000-0xffff	R	*apsAckWait Duration* + *InterframeDelay* * *NumberOfACK Frames*	M
0x0006	*InterframeDelay*	uint8	Default-0xff	RW	*apsInterFrame Delay*	M
0x0007	*NumberOfSendRetries*	uint8	0x00-0xff	R	0x03	M
0x0008	*SenderTimeout*	uint16	Default-0xffff	R	2**apsAckWait Duration* + *InterframeDelay* * *NumberOfACK Frames*	M

Id	Name	Type	Range	Acc	Default	M/O
0x0009	*ReceiverTimeout*	uint16	Default-0xffff	R	*apsAckWait Duration+ InterframeDelay +NumberOfSendRetries * NACKTimeout*	M

9.6.3.2.1.1 MaximumIncomingTransferSize Attribute

The *MaximumIncomingTransferSize* attribute specifies the maximum size, as multiple of *PartionedFrameSize*, of the application service data unit (ASDU) that can be transferred to this node in one single message transfer. The ASDU referred to is the ZCL frame, including header and payload, of any command received by a Partition cluster on the same endpoint.

9.6.3.2.1.2 MaximumOutgoingTransferSize Attribute

The *MaximumOutgoingTransferSize* attribute specifies the maximum size, as multiple of *PartionedFrameSize*, of the application service data unit (ASDU) that can be transferred from this node in one single message transfer. The ASDU referred to is the ZCL frame, including header and payload, of any command received by a Partition cluster on the same endpoint.

9.6.3.2.1.3 PartionedFrameSize Attribute

The *PartitionedFrameSize* attribute specifies the size in bytes of a partitioned frame transferred using *TransferPartitionedFrame* command. The default value for this attribute is equal to 80 bytes (0x50) because a "large frame" to be transferred using the Partition Cluster shall be partitioned into smaller *PartitionedFrameSize* frame size.

9.6.3.2.1.4 LargeFrameSize Attribute

The *LargeFrameSize* attribute specifies the size, in multiple of *PartionedFrameSize*, of a large frame to be partitioned using the Partition cluster into *PartionedFrameSize* bytes carried by *TransferPartitionedFrame* commands. The default value of this attribute should be set equal to 0x0500 (so that, given the default *PartitionedFrameSize* attribute equal to 80bytes the default large frame would be 100KB). The length in byte of the large frame to be partitioned is equal to *PartitionedFrameSize*LargeFrameSize*. In case the frame to be partitioned is not multiple of *PartitionedFrameSize*LargeFrameSize*, the last *TransferPartionedFrame* command shall be padded with zeros in order to fit in *PartionedFrameSize* length of the last *TransferPartionedFrame* command.

9.6.3.2.1.5 NumberOfACKFrame Attribute

The *NumberOfACKFrame* attribute specifies the number of partitioned frames to be received before sending a multiple acknowledge command. The proper setting of this attribute guarantee the reduction of acknowledge packet to be transmitted over the network. If *NumberOfAckFrame* attribute is set to 0x00, it indicates a non-ACK transmission. In this case, the sender would ignore the sender timeout and send the blocks continuously with *InterframeDelay* interval between each partitioned frame. In this case the receiver shall not return the *MultipleACK* after receiving the block, and the *ReceiverTimeout* and *NACKTimeout* attributes (set to the receiver) shall be also ignored.

9.6.3.2.1.6 NACKTimeout Attribute

NACKTimeout attribute specifies the maximum time, expressed in milliseconds, the receiver entity should wait after having received the last *NumberOfAckFrame* partitioned frames, before sending a *MultipleACK* command to the sender. The receiver shall transmit immediately if it receives all the partitioned frames correctly.

9.6.3.2.1.7 InterFrameDelay Attribute

The *InterFrameDelay* attribute specifies the delay in milliseconds between successive transmissions of *TransferPartionedFrame* commands. Default value for this attributes is given by the *apsInterFrameDelay*. 0x00 is not a valid value for this attribute. If the device doesn't support APS fragmentation but supports the Partition Cluster, this value shall be set to 10ms.

9.6.3.2.1.8 NumberOfSendRetries Attribute

The *NumberOfSendRetries* specifies the maximum number of retries the sender should perform in case no *MultipleACK* have been received in *SenderTimeout* time period. This attribute should be reset to the default value when a *MultipleACK* command is received.

9.6.3.2.1.9 SenderTimeout Attribute

The *SenderTimeout* attribute specifies is the time that the sender should wait for the *MultipleACK* before sending a number of *NumberOfACKFrame* of *TransferPartitionedFrame* commands again. This attribute should be reset to the default value when a *MultipleACK* command is received and started with the first block sent to the receiver.

9.6.3.2.1.10 ReceiverTimeout Attribute

The *ReceiverTimeout* attribute specifies the maximum time the receiver need to wait for a *TransferPartitionedFrame* command after the reception the first frame of the large frame to be transferred. If there will be no frames received after *ReceiverTimeout*, the receiver will exit the Partition procedure.

9.6.3.3 Commands Received

The received command IDs for the Partition cluster are listed in Table 9-30.

Table 9-30. Server Received Command IDs for the Partition Cluster

Command Identifier Field Value	Description	M/O
0x00	TransferPartitionedFrame	M
0x01	ReadHandshakeParam	M
0x02	WriteHandshakeParam	M

9.6.3.3.1 TransferPartitionedFrame Command

The *TransferPartitionedFrame* command is used to send a partitioned frame to another Partition cluster. It shall be originated by the sender device and sent to the recipient device which is expected to answer with a *MultipleACK* (as defined in 9.6.3.4.1). When the sender composes and sends to the receiver the first *TransferPartitionedFrame* command, a timer on the sender is started; this timer shall be used to check if the sender received a *MultipleACK* before *SenderTimeout* time period. The sender may wait for a *MultipleACK* after every *NumberOfACKFrame* blocks transmission. In that case the value *NumberOfACKFrame* should be set in a handshake phase. The sender will consider a successful transmission of a *NumberOfACKFrame* number of blocks if no *NACKIds* are carried by the *MultipleACK* command payload.

The *TransferPartitionedFrame* command shall be formatted as illustrated in Figure 9-8.

Figure 9-8. Format of the *TransferPartitionedFrame* Command

octets	1	1-2	Variable
Data Types	map8	uint8 or uint16	octstr
Field Name	*Fragmentation Options*	*PartitionIndicator*	*PartitionedFrame*

The *Fragmentation Options* field shall be formatted as in Figure 9-9.

Figure 9-9. Format of the *FragmentationOptions* Field

b0: 1 bit	b1: 1 bit	b2-b7: 6 bit
First block	*Indicator length*	Reserved

First Block field *b0=1* indicates that the *TransferPartitionedFrame* command carries the first block of *NumberOfACKFrame* while *b0=0* indicates that the *TransferPartitionedFrame* command doesn't carry a first block. *Indicator length* field specifies if the *PartitionIndicator* field is 1 or 2-bytes long: *b1=0* indicates that the *PartitionIndicator* is 1-byte long, *b1 = 1* indicates that the *PartitionIndicator* is 2-bytes long.

PartitionIndicator field specifies the overall number of blocks for the 1st partitioned frame (fragment), and the block index for the other fragments starting from 0x01 or 0x0001 (respectively for *b1=0* or *b1 = 1*).

The address mechanism used for the *TransferPartitionedFrame* command should not use broadcasting and it should not use multicasting.

9.6.3.3.1.1 Effect on Receipt

The receiver will start receiving *TransferPartitionedFrame* commands and start the *NACKTimeout and ReceiverTimeout* timers after the reception of the first frame related to the transaction registered by the handshake phase (*WriteHandshakeParam* command); if *NumberOfACKFrames* have been received, the Partition Cluster of the receiver will send a *MultipleACK* command with no *NACKId*. The block indexes of expected *TransferPartitionedFrame* commands that have not been received in *NACKTimeout* (*NACKIds*) will be inserted in the *MultipleACK* command returned to the sender. If there are no frames received after *ReceiverTimeout*, the receiver will exit the partition procedure. In case the receiver receives a number equal to *NumberOfACKFrame* partitioned frames it shall send the *MultipleACK* command without waiting for a *NACKTimeout* time. The receiver will also reset the *ReceiverTimeout* timer after reception of a *TransferPartitionedFrame* command.

9.6.3.3.2 ReadHandshakeParam Command

The *ReadHandshakeParam* command is used in order to read the appropriate set of parameters for each transaction to be performed by the Partition Cluster. The *Partitioned ClusterID* field identifies the specific cluster referred to the large frame that is going to be partitioned by the Partition Cluster itself. The transaction number of the specific frame to be partitioned shall be carried directly in the ZCL header.

Figure 9-10. *ReadHandshakeParam* Frame

Octets	2	2	…	2
Data Types	ClusterID	AttributeID	…	AttributeID
Field Name	Partitioned ClusterID	Attribute identifier 1	…	Attribute identifier *n*

9.6.3.3.3 WriteHandshakeParam Command

The *WriteHandshakeParam* command is used during the handshake phase in order to write the appropriate parameters for each transaction to be performed by the Partition Cluster. The *Partitioned ClusterID* field identifies the specific cluster referred to the frames that is going to be partitioned by the Partition Cluster itself. The transaction number of the specific frame to be partitioned shall be carried in the ZCL header. See 2.4.3for write attribute record format. By using the *WriteHandshakeParam* command report it is possible to write Partition Cluster attributes related to the specific large frame to be transferred using partitioning.

Figure 9-11. *WriteHandshakeParam* Frame

Octets	2	2	…	2
Data Types	ClusterID	See 2.4.3	…	See 2.4.3
Field Name	Partitioned ClusterID	Write Attribute Record 1		Write Attribute Record *n*

Figure 9-12. Format of Write Attribute Record Field

octets: 2	**1**	**Variable**
Attribute Identifier	Attribute Data Type	Attribute Data

9.6.3.4 Commands Generated

The generated command IDs for the server Partition cluster are listed in Table 9-31.

Table 9-31. Generated Command IDs for the Partition Cluster

Command Identifier Field Value	**Description**	**M/O**
0x00	*MultipleACK*	M
0x01	*ReadHandshakeParamResponse*	M

9.6.3.4.1 MultipleACK Command

The receiver shall return the *MultipleACK* command when receiving a number equal to *NumberOfACKFrame TransferPartitionedFrame* commands (partitioned frames) or when *NACKTimeout* expires. The *MultipleACK* command will carry no *NACKId* in the payload if *NumberOfACKFrame TransferPartitionedFrame* commands are received. The sender may wait for a *MultipleACK* command after every *NumberOfACKFrame* blocks transmission. The *MultipleACK* command shall be formatted as illustrated in Figure 9-13.

Figure 9-13. Format of the *MultipleACK* Command

Octets	1	1-2	1-2	1-2	1-2
Data Types	map8	uint8 or uint16	uint8 or uint16		uint8 or uint16
Field Name	*ACK Options*	*FirstFrameID*	*NACKId*	...	*NACKId*

The *ACKOptions* payload fields shall be formatted as illustrated in Figure 9-14.

Figure 9-14. Format of the *ACK Options* Field

b0: 1 bit	**b1-b7: 7 bit**
NACKId length	Reserved

NACKId length specifies if the *NACKId*, corresponding to the *PartitionIndicator* (*NACKIds* carried in this command are the values of the "PartitionIndicator" field), and the *FirstFrameID* are 1 or 2 bytes long: *b0=0* indicates that the *NACKIds* and the *FirstFrameID*, are 1-byte long, *b0 = 1* indicates that the *NACKIds* and the *FirstFrameID*, are 2-bytes long.

FirstFrameID field indicates the first partition frame (block) index of the current overall *NumberOfACKFrame* blocks the *MultipleACK* refers to. It is used in order to identify the set of *NumberOfACKFrame* the *MultipleACK* command refers to.

NACKId fields represent the ID of partitioned frame that have not been received yet after *NACKTimeout*.

9.6.3.4.1.1 Effect on Receipt

After sending a number of *TransferPartitionedFrame* commands equal to *NumberOfACKFrame* (Number of acknowledged frames) the sender will wait for a *MultipleACK*: a successful transmission is indicated by a *MultipleACK* command with no *NACKId* fields carried. The sender shall stop sending the next *NumberOfACKFrame* blocks until it receives a *MultipleACK* command reporting a successful transmission.

When the sender successfully sends the current *NumberOfACKFrame* blocks and receives a *MultipleACK* command with no *NACKId* fields, the Partition Cluster should proceed to send the next *NumberOfACKFrame* set of blocks of the, large frame to be transmitted, until all the set of blocks have been sent out. The partition parameters such as *NumberOfACKFrame* may be tuned after sending out the current *NumberOfACKFrame*, set of blocks (e.g., the value of *NumberOfACKFrame* may be decreased after retransmissions of many *TransferPartitionedFrame* commands of a previous transaction).

In case the receiver does need to send out several *MultipleACKs* to the sender, it should not send out a next one until completing the reception of all blocks indicated in the *NACKId* fields of the previous *MultipleACK*. The sender should receive *MultipleACK* command by sender timeout (this timeout specifies how long to wait for a *MultipleACK*); if no *MultipleACK* command is received the sender will retransmit the *TransferPartitionedFrame* commands up to a maximum number of retries (in order to optimize the protocol the sender may reduce also the *NumberOfACKFrame* value by using the writing command defined in the handshake phase); if the sender doesn't receive any *MultipleACK* after maximum number of retries it will exit the partition procedure and the *TransferFrameUsingPartitionCluster* response will notify the error in the partition procedure; otherwise, if *MultipleACK* is received carrying some *NACK IDs*, the sender will reset the sender timeout and the max number of retries and resend the no acknowledged *TransferPartitionedFrame* commands up to max number of retries until a *MultipleACK* with no *NACK* is received (success in the partition transaction) or the *SenderTimeout* expires (in case no *MultipleACK* commands are received) or max number of retries reached (in case *MultipleACK* commands are received but still with *NACKIds*).

The *SenderTimeout* is equal to *2*apcAckWaitDuration + InterframeDelay * NumberOfACKFrames*.

9.6.3.4.2 ReadHandshakeParamResponse Command

The *ReadHandshakeParamResponse* command is used in order to response to the corresponding *ReadHandshakeParam* command in order to communicate the appropriate set of parameters configured for each transaction to be performed by the Partition Cluster. The *Partitioned ClusterID* field identifies the specific cluster referred to the large frame that is going to be partitioned by the Partition Cluster itself. The transaction number of the specific frame to be partitioned shall be carried directly in the ZCL header. The rRead Attribute status record field is the same as defined for the ZCL (see 2.4.2.1).

Figure 9-15. *ReadHandshakeParamResponse* Frame

octets	2	Variable	…	Variable
Data Types	ClusterID	See 2.4.2.1	…	See 2.4.2.1
Field Name	Partitioned ClusterID	Read attribute status record 1	…	Read attribute status record *n*

Figure 9-16. Format of Read Attribute Status Record Field

octets: 2	2	0/1	0/Variable
Attribute Identifier	Status	Attribute Data Type	Attribute Data

9.6.4 Client

9.6.4.1 Attributes

None

9.6.4.2 Command Received

The client receives the cluster specific commands detailed in 9.6.3.4, as required by application profiles.

9.6.4.3 Command Generated

The client generates the cluster specific commands detailed in 9.6.3.3 as required by application profiles.

9.6.5 General Use of Partition Cluster

The Partition cluster may be used by multiple clusters defined in a single application object. In order to perform the recognition of multiple partitioned frames associated to a specific cluster and reconstruct a partitioned large frame the Partition cluster shall maintain an internal table similar to the one presented in Table 9-32: Each large frame to be partitioned can be identified by the ClusterID and the ZCL transaction sequence number.

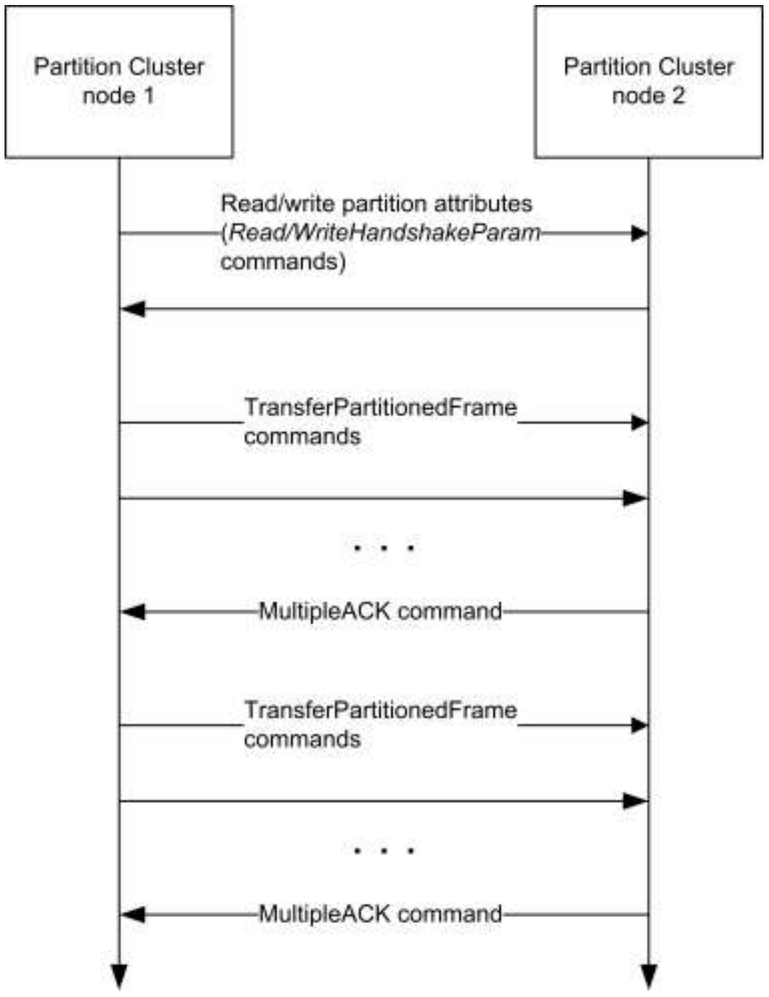
The specific clusters using the Partition cluster to transfer large frame shall subscribe to the registration table writing an entry for each frame to be partitioned with the Partition Cluster attributes fields specified in 9.6.3.2. This entry shall be cancelled by the Partition Cluster when the frame is correctly transferred or the partitioning procedure exited with errors. The entries of this table should be inserted during the handshake phase, i.e., in the sender when the *WriteHandshakeParam* command is generated and in the receiver when the *WriteHandshakeParam* command is received.

The partitioned frames generated from the partitioning of a large frame shall use the ZCL transaction sequence number inserted in the ZCL header of the large frame for each small partitioned frame (transferred using the *TransferPartitionedFrame* commands) in order to identify the proper fragment if multiple partitions are running on the same endpoint with large frames carrying the same ClusterIDs.

Table 9-32. Registration Table of Clusters Using the Partition Cluster

ClusterID	Transaction sequence number	*Partition Cluster attributes*
Registered cluster ID	Transaction sequence number (of the packet to be partitioned) through the Partition cluster	Attributes that are written using the *WriteHandshakeParam* command

Figure 9-17. Example of Partition Cluster Use


Partition Cluster node 1
Partition Cluster node 2
Read/write partition attributes (*Read/WriteHandshakeParam* commands)
TransferPartitionedFrame commands
. . .
MultipleACK command
TransferPartitionedFrame commands
. . .
MultipleACK command

9.7 11073 Protocol Tunnel

9.7.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The 11073 Protocol Tunnel cluster provides the commands and attributes required to tunnel the 11073 protocol. The server cluster receives 11073 APDUs and the client cluster generates 11073 APDUs, thus it is necessary to have both server and client on an endpoint to tunnel 11073 messages in both directions.

Commands and attributes are provided for establishing, querying the status of, and removing an 11073 tunnel connection between two devices.

Devices that support this cluster shall also comply with the ISO/IEEE 11073-20601 standard for Personal Health Device Communication [H1] and the applicable ISO/IEEE 11073 device specialization documents [H2] – [H12].

Typical usage of the 11073 Protocol Tunnel cluster is illustrated in Figure 9-18.

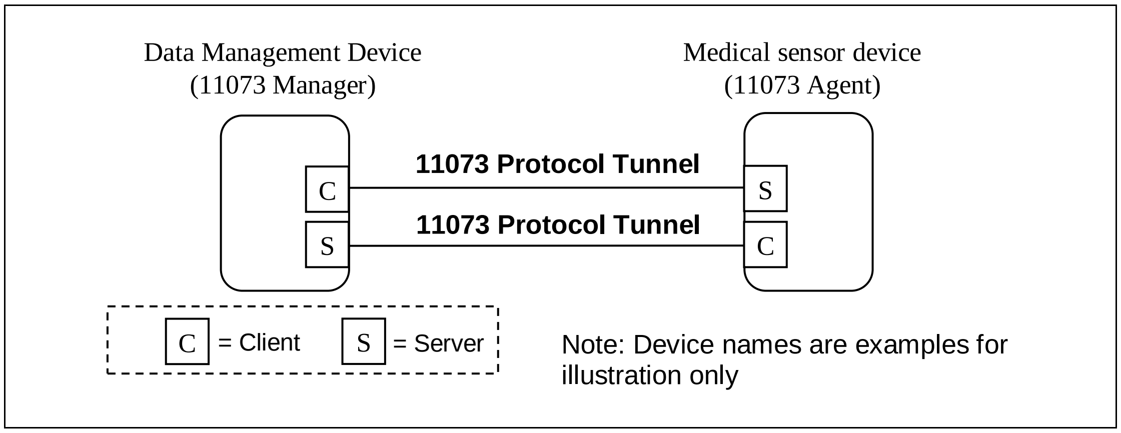


Figure 9-18 Typical Usage of the 11073 Protocol Tunnel cluster

Note that all 11073 protocol tunnel cluster specific commands are generated by the client and received by the server. A typical sequence of events to initiate an 11073 interaction might be:

- DMD transmits Connect Request command to sensor (client→server)
- Sensor responds with a Connect Status Notification command with status CONNECTED (client→server)
- Sensor and DMD carry out an 11073 layer interaction by exchanging Transfer APDU commands in each direction (client→server in each case)

9.7.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

9.7.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	T11073	Type 1 (client to server)

9.7.1.3 Cluster Identifiers

Identifier	Name
0x0614	11073 Protocol Tunnel

9.7.2 Server

9.7.2.1 Dependencies

Any endpoint that supports the 11073 Protocol Tunnel server cluster shall also support the Generic Tunnel server cluster (see 9.2).

The value of the *ProtocolAddress* attribute of the associated Generic Tunnel server cluster shall be set equal to the system ID of the 11073 device represented on that endpoint (see [H1]). The system ID, represented as a octet string, shall be 8 octets in Big Endian order.

The value of the *MaximumIncomingTransferSize* attribute and the value of the *MaximumOutgoingTransferSize* attribute of the associated Generic Tunnel server cluster shall be set equal to or greater than the maximum APDU size specified in the applicable ISO/IEEE 11073 device specialization document ([H2] – [H12]).

9.7.2.2 Attributes

The 11073 Protocol Tunnel server cluster contains the attributes shown in Table 9-33:

Table 9-33 – Attributes of the 11073 Protocol Tunnel server cluster

ID	Name	Type	Range	Acc	Default	M/O
0x0000	*DeviceIDList*	array of uint16	Any valid	R	0xffff	O
0x0001	*ManagerTarget*	IEEE address	Any valid IEEE address	R	-	O
0x0002	*ManagerEndpoint*	uint8	0x01-0xff	R	-	O
0x0003	*Connected*	bool	TRUE / FALSE	R	FALSE	O
0x0004	*Preemptible*	bool	TRUE / FALSE	R	TRUE	O
0x0005	*IdleTimeout*	uint16	0x0001 – 0xffff	R	0x0000	O

Although the *ManagerTarget*, *ManagerEndpoint*, *Connected*, *Preemptible* and *IdleTimeout* attributes are listed above as optional, if any one of them is implemented then all of them shall be implemented, and also reception of the connect and disconnect request commands shall be implemented, and the 11073 Protocol Tunnel client cluster implemented on the same endpoint shall implement transmission of the connect status notification command.

9.7.2.2.1 DeviceIDList attribute

The *DeviceIDList* attribute specifies all devices supported behind a single instance of the11073 tunnel, on a single endpoint. It allows for discovering the functionality of 11073 devices, e.g. prior to establishment of an 11073 tunnel. The *DeviceIDList* attribute can be read using generic ZCL commands.

For a multifunction device as defined in [Z6], the *DeviceIDList* attribute is mandatory and shall contain a complete list of supported Device IDs (as defined in [Z6]) supported by the single instance of the *11073 Protocol Tunnel* on this particular endpoint. The DeviceID contained in the Simple Descriptor of the multifunction sensor shall contain the value corresponding to the multifunction device itself (see [Z6]).

For all other devices types, the *DeviceIDList* attribute is optional. If implemented, it shall contain the single DeviceID allocated to that device (see [Z6]).

9.7.2.2.2 ManagerTarget attribute

The *ManagerTarget* attribute specifies the IEEE address of the currently or most recently connected Data Management device.

9.7.2.2.3 ManagerEndpoint attribute

The *ManagerEndpoint* attribute specifies the endpoint used by the currently or most recently connected Data Management device.

9.7.2.2.4 Connected attribute

The *Connected* attribute specifies whether or not the 11073 tunnel on this endpoint is currently connected.

If this attribute takes the value TRUE, then the tunnel is currently connected.

If this attribute takes the value FALSE, then the tunnel is not currently connected.

Whenever the value of this attribute changes the 11073 layer shall be informed via the transport connected and transport disconnected indications.

9.7.2.2.5 Preemptible attribute

The *Preemptible* attribute specifies whether or not the current connection can be disconnected by a Data Management device other than by the one currently connected.

If this attribute takes the value TRUE, then a disconnect request from a device other than the Data Management device indicated by the *ManagerTarget* attribute shall be accepted and if the 11073 tunnel on this endpoint is currently connected then it shall become disconnected, and a connect status notification command with status DISCONNECTED shall be sent to the currently connected Data Management device.

If this attribute takes the value FALSE, then a disconnect request from a device other than the Data Management device indicated by the *ManagerTarget* attribute shall be rejected and a connect status notification command with status NOT_AUTHORIZED sent to the requester.

9.7.2.2.6 Idle timeout attribute

The *Idle Timeout* attribute specifies the inactivity time in minutes which the Data Management device will wait without transmitting or receiving any tunneled frames to or from the connected target, before it disconnects the connection.

If the Data Management device does not intend to timeout this connection after a specific idle period then this attribute shall take the value 0xffff.

If the indicated timeout period passes with no data on the 11073 tunnel, then the agent device shall set its *Connected* attribute to FALSE and a connect status notification command with status DISCONNECTED shall be sent to the currently connected Data Management device. In order to continue to use the tunnel, the agent device shall send the Data Management device a further connect status notification command with status RECONNECT_REQUEST, and wait for the Data Management device to respond.

9.7.2.3 Commands Received

The cluster specific commands received by the 11073 Protocol Tunnel server cluster are listed in Table 9-34:

Table 9-34 – Command IDs for the 11073 protocol tunnel cluster

Command identifier field value	Description	Mandatory/Optional
0x00	Transfer APDU	M
0x01	Connect request	O

Command identifier field value	Description	Mandatory/Optional
0x02	Disconnect request	O
0x03	Connect status notification	O

Although the connect request and disconnect commands are listed above as optional, if reception of either of them is implemented then reception of both of them shall be implemented, and also the *ManagerTarget*, *ManagerEndpoint*, *Connected*, *Preemptible* and *IdleTimeout* attributes shall be implemented, and the 11073 Protocol Tunnel client cluster implemented on the same endpoint shall implement transmission of the connect status notification command.

Although reception of the connect status notification command is listed above as optional, if reception of this command is implemented then also the connect request command shall be implemented by the 11073 Protocol Tunnel server on the same endpoint.

9.7.2.3.1 Transfer APDU Command

The Transfer APDU command payload shall be formatted as illustrated in Figure 9-19:

Bits	Variable
Data Type	long octet string
Field Name	APDU

Figure 9-19 – Transfer APDU payload

The APDU field is of variable length and is a 11073 APDU as defined in the ISO/IEEE 11073 standard [H1].

9.7.2.3.1.1 When generated

This command is generated when an 11073 network layer wishes to transfer an 11073 APDU across a tunnel to another 11073 network layer.

The most stringent reliability characteristic of a given transport technology is “Best” reliability. Note - For ZigBee, this corresponds to use of APS-ACKs.

The least stringent reliability characteristic of a given transport technology is “Good” reliability. Note - For ZigBee, this corresponds to no use of APS-ACKs.

The application is responsible for transmitting at a reliability level appropriate for each frame.

This command shall always be transmitted with the disable default response bit in the ZCL frame control field set to 1.

9.7.2.3.1.2 Effect on Receipt

On receipt of this command, a device shall process the 11073 APDU as specified in [H1] and the applicable device specialization [H2] to [H12]

9.7.2.3.2 Connect Request Command

The Connect Request command payload shall be formatted as illustrated in Figure 2.

Octets	1	2	8	1
Data Type	map8	uint16	IEEE address	uint8
Field Name	Connect control	Idle timeout	Manager target	Manager endpoint

Figure 9-20 – Connect Request command payload

9.7.2.3.2.1 Connect control

The *connect control* field shall be formatted as illustrated in Figure 9-21:

Bit	0	1-7
Field Name	Preemptible	Reserved

Figure 9-21 – Connect control field format

The *Preemptible* bit shall indicate whether or not this connection can be removed by a different Data Management device.

9.7.2.3.2.2 Idle timeout

The *idle timeout* field shall indicate the inactivity time in minutes which the Data Management device will wait without receiving any tunneled frames from the connected target, before it disconnects the connection.

9.7.2.3.2.3 Manager target

The *Manager target* field shall indicate the IEEE address of the Data Management device transmitting this frame.

9.7.2.3.2.4 Manager endpoint

The *Manager endpoint* field shall indicate the source endpoint from which the Data Management device is transmitting this frame.

9.7.2.3.2.5 When generated

This command is generated when a Data Management device wishes to connect to an 11073 agent device.

This may be in response to receiving a connect status notification command from that agent device with the connect status field set to RECONNECT_REQUEST.

9.7.2.3.2.6 Effect on Receipt

On receipt of this command, a device shall first check if it is already connected by examining its *Connected* attribute.

If the tunnel is already connected then the device shall generate a connect status notification command with status set to ALREADY_CONNECTED and transmit it to the sender of this connect request frame. No other attributes shall be affected, and no further processing shall be carried out.

If the tunnel is not currently connected then the device shall copy the preemptible bit of connect control field into the preemptible attribute, the idle timeout value into the idle timeout attribute, the manager target value into the *ManagerTarget* attribute and the manager endpoint value into the *ManagerEndpoint* attribute.

It shall set the connected attribute to TRUE, and generate a connect status notification command with status set to CONNECTED and transmit it to the sender of this connect request frame.

Finally, if the idle timeout field is set to a value other than 0xffff, the device shall set a timer for the timeout time indicated. This timer shall be restarted at any time that data is transmitted or received over the tunnel. If the timer expires then the device shall set the *Connected* attribute to FALSE and a connect status notification command with status DISCONNECTED shall be sent to the currently connected Data Management device. In order to continue to use the tunnel, the agent device shall send the Data Management device a further connect status notification command with status RECONNECT_REQUEST, and wait for the Data Management device to respond.

9.7.2.3.3 Disconnect Request Command

The Disconnect Request command payload shall be formatted as illustrated in Figure 9-22.

Octets	8
Data Type	IEEE address
Field Name	Manager IEEE address

Figure 9-22 – Disconnect Request command payload

9.7.2.3.3.1 Manager IEEE address

The *Manager IEEE address* field shall indicate the IEEE address of the Data Management device transmitting this frame.

9.7.2.3.3.2 When generated

This command is generated when a Data Management device wishes to disconnect a tunnel connection existing on an agent device.

9.7.2.3.3.3 Effect on Receipt

On receipt of this command, a device shall first check if it is already connected by examining its *Connected* attribute.

If it is not currently connected then the device shall generate a connect status notification command with status set to DISCONNECTED and transmit it to the sender of this disconnect request frame. No other attributes shall be affected, and no further processing shall be carried out.

If it is currently connected then the device shall check whether the requesting device is authorized to remove this connection. A device is authorized to remove the connection if the value of the manager IEEE address field is the same as the value in the *ManagerTarget* attribute or if the *Preemptible* attribute is set to TRUE.

If the requester is not authorized then the device shall generate a connect status notification command with status set to NOT_AUTHORIZED and transmit it to the sender of this disconnect request frame. No other attributes shall be affected, and no further processing shall be carried out.

If the requester is authorized then the device shall initiate disconnection. A short period of time is permitted in order to allow the higher layer to finalize its activities, but within 12 seconds the device shall generate a connect status notification command with status set to DISCONNECTED and transmit it to the target indicated in the *ManagerTarget* attribute. The *Connected* attribute shall be set to FALSE and the tunnel shall be disconnected. The device shall now generate a further connect status notification command with status set to DISCONNECTED and transmit it to the sender of this disconnect request frame.

9.7.2.3.4 Connect Status Notification Command

The Connect Status Notification command payload shall be formatted as illustrated in Figure 9-23.

Octets	1
Data Type	enum8
Field Name	Connect status

Figure 9-23 – Connect Status Notification command payload

9.7.2.3.4.1 Connect Status

The *connect status* field shall be set to one of the values in Table 9-35:

Table 9-35 – Connect status values

Value	Designation	Description
0x00	DISCONNECTED	Indicates that this agent device has been disconnected from the tunnel.
0x01	CONNECTED	Indicates that this agent device has been connected to the tunnel.
0x02	NOT_AUTHORIZED	Indicates that a request to disconnect the tunnel is not authorized from this requester at this time.
0x03	RECONNECT_REQUEST	Indicates that the agent device wishes the Data Management device to reconnect the tunnel.
0x04	ALREADY_CONNECTED	Indicates that the request to connect this tunnel has failed as the agent device is already connected.

9.7.2.3.4.2 When generated

This command is generated by an agent device in response to a connect request command, disconnect command, or in response to some other event that causes the tunnel to become connected or disconnected.

It is also sent by the agent device to request the Data Management device to reconnect a tunnel.

9.7.2.3.4.3 Effect on Receipt

On receipt of this command, a device shall be informed of the new status of the tunnel connection or of its attempt to modify the status of the connection.

If the connect status field takes the value RECONNECT_REQUEST then, depending on available resources being available, the Data Management device should attempt to reconnect the tunnel by generating a connect request command and transmitting it to the agent device sending this connect status notification command.

9.7.2.4 Commands Generated

No cluster specific commands are generated by the server cluster.

9.7.3 Client

9.7.3.1 Dependencies

Any endpoint that supports the 11073 Protocol Tunnel client cluster shall also support the Generic Tunnel client cluster (see 9.2).

9.7.3.2 Attributes

The client cluster has no attributes.

9.7.3.3 Commands Received

The client does not receive any cluster specific commands.

9.7.3.4 Commands Generated

The cluster specific commands generated by the client cluster are listed in 9.7.2.3.

In order to reduce the burden on implementations, some commands and attributes are conditionally mandated, as follows:

- Transmission of the transfer APDU command is mandatory.
- Transmission of the connect request and disconnect request commands is optional unless specified otherwise.
- If the 11073 Protocol Tunnel server cluster implemented on the same endpoint implements any of the *ManagerTarget*, *ManagerEndpoint*, *Connected*, *Preemptible* and *IdleTimeout* attributes, or implements reception of the connect request or disconnect request commands, then transmission of the connect status notification command is mandatory.
- Transmission of non-cluster specific commands to manipulate attributes is optional unless specified otherwise.

CHAPTER 10 SMART ENERGY

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

10.1 General Description

10.1.1 Introduction

The clusters specified in this chapter are for use typically in Smart Energy applications with associated security controls at the application layer. These clusters may be used in any application domain.

10.1.2 Cluster List

This section lists the clusters specified in this chapter and gives examples of typical usage for the purpose of clarification. The clusters specified in this chapter are listed in Table 10-1.

Table 10-1. Smart Energy Clusters

Cluster ID	Cluster Name	Description
0x0700	Price	Commands and attributes for reporting price
0x0701	Demand Response and Load Control	Commands and attributes for providing demand response and load control of devices
0x0702	Metering	Commands and attributes for reporting metering data
0x0703	Messaging	Commands and attributes for sending messages to devices
0x0704	Tunneling	Commands and attributes for establishing and using a tunnel between two devices
0x0800	Key Establishment	Commands and attributes for application level security establishment

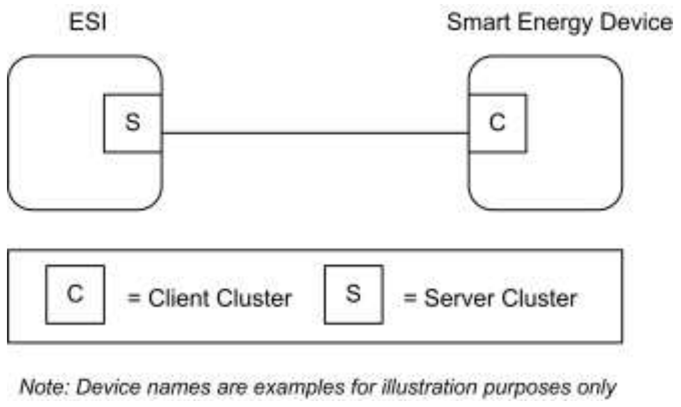
10.2 Price

10.2.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The Price Cluster provides the mechanism for communicating Gas, Energy, or Water pricing information within the premises. This pricing information is distributed to the ESI from either the utilities or from regional energy providers. The ESI conveys the information (via the Price Cluster mechanisms) to both Smart Energy devices in secure method and/or optionally conveys it anonymously in an unsecure to very simple devices that may not be part of the Smart Energy network. The mechanism for sending anonymous information is called the Anonymous Inter-PAN transmission mechanism.

Figure 10-1. Price Cluster Client Server Example


ESI
Smart Energy Device
S
C
C
= Client Cluster
S
= Server Cluster
Note: Device names are examples for illustration purposes only

Please note the ESI is defined as the Server due to its role in acting as the proxy for upstream price management systems and subsequent data stores.

10.2.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

10.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	SEPR	Type 1 (client to server)

10.2.1.3 Cluster Identifiers

Identifier	Name
0x0700	Price

10.2.2 Server

10.2.2.1 Dependencies

- Events carried using this cluster include a timestamp with the assumption that target devices maintain a real time clock. Devices can acquire and synchronize their internal clocks via the ZCL Time server.
- If a device does not support a real time clock it is assumed that the device will interpret and utilize the “Start Now” value within the Time field.
- Anonymous Inter-PAN transmission mechanism.

10.2.2.2 Attributes

For convenience, the attributes defined in this cluster are arranged into sets of related attributes; each set can contain up to 256 attributes. Attribute identifiers are encoded such that the most significant octet specifies the attribute set and the least significant octet specifies the attribute within the set. The currently defined attribute sets are listed in Table 10-2.

Note: Price Cluster Attribute Set 0x03 in this revision of this specification is provisional and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

Table 10-2. Price Cluster Attribute Sets

Attribute Set Identifier	Description
0x00	Tier Label
0x01	Block Threshold
0x02	Block Period
0x03	Commodity
0x04	Block Price Information
0x07	Billing Information Set

10.2.2.2.1 Tier Label Set

Note: Tier Label Set 0x06-0x0E in this revision of this specification are provisional and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

Table 10-3. Tier Label Attribute Set

Id	Name	Type	Length	Access	Default	M/O
0x0000	*Tier1PriceLabel*	octstr	1 to 13 Octets	RW	“Tier 1”	O
0x0001	*Tier2PriceLabel*	octstr	1 to 13 Octets	RW	“Tier 2”	O
0x0002	*Tier3PriceLabel*	octstr	1 to 13 Octets	RW	“Tier 3”	O
0x0003	*Tier4PriceLabel*	octstr	1 to 13 Octets	RW	“Tier 4”	O
0x0004	*Tier5PriceLabel*	octstr	1 to 13 Octets	RW	“Tier 5”	O
0x0005	*Tier6PriceLabel*	octstr	1 to 13 Octets	RW	“Tier 6”	O
0x0006	*Tier7PriceLabel*	octstr	1 to 13 Octets	R	“Tier 7”	O
0x0007	*Tier8PriceLabel*	octstr	1 to 13 Octets	R	“Tier 8”	O
0x0008	*Tier9PriceLabel*	octstr	1 to 13 Octets	R	“Tier 9”	O
0x0009	*Tier10PriceLabel*	octstr	1 to 13 Octets	R	“Tier 10”	O
0x000A	*Tier11PriceLabel*	octstr	1 to 13 Octets	R	“Tier 11”	O
0x000B	*Tier12PriceLabel*	octstr	1 to 13 Octets	R	“Tier 12”	O
0x000C	*Tier13PriceLabel*	octstr	1 to 13 Octets	R	“Tier 13”	O
0x000D	*Tier14PriceLabel*	octstr	1 to 13 Octets	R	“Tier 14”	O
0x000E	*Tier15PriceLabel*	octstr	1 to 13 Octets	R	“Tier 15”	O

10.2.2.2.1.1 TierNPriceLabel Attributes

The TierNPriceLabel attributes provide a method for utilities to assign a label to the Price Tier declared within the Publish Price command. The TierNPriceLabel attributes are a ZCL Octet String field capable of storing a 12 character string (the first octet indicates length) encoded in the UTF-8 format. Example Tier Price Labels are “Normal”, “Shoulder”, “Peak”, “Real Time,” and “Critical”.

10.2.2.2.2 Block Threshold Set

The set of attributes shown in Table 10-4 provides remote access to the Price server Block Thresholds. Block Threshold values are crossed when the CurrentBlockPeriodConsumptionDelivered attribute value is greater than a BlockNThreshold attribute. The number of block thresholds is indicated by the Number of Block Thresholds field in the associated Publish Price command. The number of blocks is one greater than the number of thresholds.

Table 10-4. Block Threshold Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0000	*Block1Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0001	*Block2Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0002	*Block3Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0003	*Block4Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0004	*Block5Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0005	*Block6Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0006	*Block7Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0007	*Block8Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0008	*Block9Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0009	*Block10Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x000A	*Block11Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x000B	*Block12Threshold*	uint48	0x000000000000 to xFFFFFFFFFFFF	R	-	O
0x000C	*Block13Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x000D	*Block14Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x000E	*Block15Threshold*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O

10.2.2.2.2.1 BlockNThreshold

Attributes Block1Threshold through Block15Threshold represent the block threshold values for a given period (typically the billing cycle). These values may be updated by the utility on a seasonal or annual basis. The thresholds are established such that crossing the threshold of energy consumption for the present block activates the next higher block, which can affect the energy rate in a positive or negative manner. The values are absolute and always increasing. The values represent the threshold at the end of a block. The Unit of Measure will be based on the fields defined in the Publish Price command, the formatting being defined by attributes within the Block Period attribute set.

10.2.2.2.3 Block Period Set

The set of attributes shown in Table 10-5 provides remote access to the Price server Block Threshold period (typically the billing cycle) information.

Table 10-5. Block Period Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*StartofBlockPeriod*	UTC	-	R	-	O
0x0001	*BlockPeriodDuration*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x0002	*ThresholdMultiplier*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x0003	*ThresholdDivisor*	uint24	0x000000 to 0xFFFFFF	R	-	O

10.2.2.2.3.1 StartofBlockPeriod Attribute

The StartofBlockPeriod attribute represents the start time of the current block tariff period. A change indicates that a new Block Period is in effect (see sub-clause 10.2.4.3 for further details).

10.2.2.2.3.2 BlockPeriodDuration Attribute

The BlockPeriodDuration attribute represents the current block tariff period duration in minutes. A change indicates that only the duration of the current Block Period has been modified. A client device shall expect a new Block Period following the expiration of the new duration.

10.2.2.2.3.3 ThresholdMultiplier Attribute

ThresholdMultiplier provides a value to be multiplied against Threshold attributes. If present, this attribute must be applied to all Block Threshold values to derive values that can be compared against the CurrentBlockPeriodConsumptionDelivered attribute within the Metering cluster (see 10.4.2.2.1.13). This attribute must be used in conjunction with the ThresholdDivisor attribute. An attribute value of zero shall result in a unitary multiplier (0x000001).

10.2.2.2.3.4 ThresholdDivisor Attribute

ThresholdDivisor provides a value to divide the result of applying the ThresholdMultiplier attribute to Block Threshold values to derive values that can be compared against the CurrentBlockPeriodConsumptionDelivered attribute within the Metering cluster (see 10.4.2.2.1.13). This attribute must be used in conjunction with the ThresholdMultiplier attribute. An attribute value of zero shall result in a unitary divisor (0x000001).

10.2.2.2.4 Commodity Set

The set of attributes shown in Table 10-6 represents items that are associated with a particular commodity.

Note: With the exception of the Standing Charge attribute, the Commodity Attribute Set in this revision of this specification is provisional and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

Table 10-6. *Commodity* Attribute Set

Id	Name	Type	Range	Acc	Default	M/O
0x0000	*CommodityType*	enum8	0x00 to 0xFF	R	-	O
0x0001	*StandingCharge*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0002	*ConversionFactor*	uint32	0x00000000 to 0xFFFFFFFF	R	0x10000000	O
0x0003	*ConversionFactorTrailingDigit*	map8		R	0x70	O
0x0004	*CalorificValue*	uint32	0x00000000 to 0xFFFFFFFF	R	0x2625A00	O
0x0005	*CalorificValueUnit*	enum8		R	0x1	O
0x0006	*CalorificValueTrailingDigit*	map8		R	0x60	O

10.2.2.2.4.1 CommodityType Attribute

CommodityType provides a label for identifying the type of pricing server present. The attribute is an enumerated value representing the commodity. The defined values are represented by the non-mirrored values (0-127) in the MeteringDeviceType attribute enumerations (refer to Table 10-42).

10.2.2.2.4.2 Standing Charge Attribute

The value of the Standing Charge is a daily fixed charge associated with supplying the commodity, measured in base unit of Currency with the decimal point located as indicated by the Trailing Digits field of a Publish Price command (see sub-clause 10.2.2.4.1). A value of 0xFFFFFFFF indicates field not used.

10.2.2.2.4.3 ConversionFactor Attribute

The conversion factor is used for gas meter and takes into account changes in the volume of gas based on temperature and pressure. The ConversionFactor attribute represents the current active value. The ConversionFactor is dimensionless. The default value for the ConversionFactor is 1, which means no conversion is applied. A price server can advertise a new/different value at any time.

10.2.2.2.4.4 ConversionFactorTrailingDigit Attribute

An 8-bit bitmap used to determine where the decimal point is located in the ConversionFactor attribute. The most significant nibble indicates the number of digits to the right of the decimal point. The least significant nibble is reserved. The ConversionFactorTrailingDigit attribute represents the current active value.

10.2.2.2.4.5 CalorificValue Attribute

The amount of heat generated when a given mass of fuel is completely burned. The CalorificValue is used to convert the measured volume or mass of gas into kWh. The CalorificValue attribute represents the current active value.

10.2.2.2.4.6 CalorificValueUnit Attribute

This attribute defines the unit for the CalorificValue. This attribute is an 8-bit enumerated field. The values and descriptions for this attribute are listed in Table 10-7 below. The CalorificValueUnit attribute represents the current active value.

Table 10-7. Values and Descriptions for the *CalorificValueUnit* Attribute

Values	Description
0x01	MJ/m3
0x02	MJ/kg

10.2.2.2.4.7 CalorificValueTrailingDigit Attribute

An 8-bit bitmap used to determine where the decimal point is located in the CalorificValue attribute. The most significant nibble indicates the number of digits to the right of the decimal point. The least significant nibble is reserved. The CalorificValueTrailingDigit attribute represents the current active value.

10.2.2.2.5 Block Price Information Set

The set of attributes shown in Table 10-8 provide remote access to the block prices. The Block Price Information attribute set supports Block and combined Tier-Block pricing, the number of blocks is one greater than the number of block thresholds defined in the Pricing cluster.

Table 10-8. Block Price Information Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*NoTierBlock1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0001	*NoTierBlock2Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0002	*NoTierBlock3Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x000*N*	*NoTierBlock*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x000F	*NoTierBlock16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0010	*Tier1Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0011	*Tier1Block2Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0012	*Tier1Block3Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x001*N*	*Tier1Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x001F	*Tier1Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0020	*Tier2Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x002*N*	*Tier2Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x002F	*Tier2Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0030	*Tier3Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x003*N*	*Tier3Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x003F	*Tier3Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0040	*Tier4Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x004*N*	*Tier4Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x004F	*Tier4Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0050	*Tier5Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x5*N*	*Tier5Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O

0x5F	*Tier5Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x60	*Tier6Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x6*N*	*Tier6Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x6F	*Tier6Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x70	*Tier7Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x7*N*	*Tier7Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x7F	*Tier7Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x80	*Tier8Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x8*N*	*Tier8Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x8F	*Tier8Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x90	*Tier9Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x9*N*	*Tier9Block*N+1*Price* ...	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x9F	*Tier9Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xA0	*Tier10Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xA*N*	*Tier10Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xAF	*Tier10Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xB0	*Tier11Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xB*N*	*Tier11Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xBF	*Tier11Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xC0	*Tier12Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xC*N*	*Tier12Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O

0xCF	*Tier12Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xD0	*Tier13Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xD*N*	*Tier13Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xDF	*Tier13Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xE0	*Tier14Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xE*N*	*Tier14Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xEF	*Tier14Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xF0	*Tier15Block1Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xF*N*	*Tier15Block*N+1*Price…*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xFF	*Tier15Block16Price*	uint32	0x00000000 to 0xFFFFFFFF	R	-	O

10.2.2.2.5.1 TierNBlockNPrice Attributes

Attributes PriceNoTierBlock1 through PriceTier15Block16 represent the price of Energy, Gas, or Water delivered to the premises (i.e. delivered to the customer from the utility) at a specific price tier as defined by a TOU schedule, Block Threshold or a real time pricing period. If optionally provided, attributes shall be initialized prior to the issuance of associated Publish Price commands (see sub-clause 10.2.2.4.1). The expected practical limit for the number of PriceTierNBlockN attributes supported is 32. The Unit of Measure, Currency and Trailing Digits that apply to this attribute should be obtained from the appropriate fields in a Publish Price command.

10.2.2.2.6 Billing Information Attribute Set

The set of attributes shown in Table 10-9 provides remote access to the Price server Billing information.

Table 10-9. Billing Information Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*CurrentBillingPeriodStart*	UTC	0x00000000 to 0xFFFFFFFF	R	-	O
0x0001	*CurrentBillingPeriodDuration*	uint24	0x000000 to 0xFFFFFF	R	-	O

10.2.2.2.6.1 CurrentBillingPeriodStart Attribute

The CurrentBillingPeriodStart attribute represents the start time of the current billing period.

10.2.2.2.6.2 CurrentBillingPeriodDuration Attribute

The CurrentBillingPeriodDuration attribute represents the current billing period duration in minutes.

10.2.2.3 Commands Received

The server side of the Price cluster is capable of receiving the commands listed in Table 10-10.

Table 10-10. Received Command IDs for the Price Cluster

Command Identifier Field Value	Description	M/O
0x00	*Get Current Price*	M
0x01	*Get Scheduled Prices*	O
0x02	*Price Acknowledgement*	M for 1.1 and later devices
0x03	*Get Block Period(s)*	O
0x04	*GetConversionFactor*	O
0x05	*GetCalorificValue*	O

10.2.2.3.1 Get Current Price Command

This command initiates a Publish Price command (see sub-clause 10.2.2.4.1) for the current time.

10.2.2.3.1.1 Payload Format

The payload of the Get Current Price command is formatted as shown in Figure 10-2.

Figure 10-2. The Format of the *Get Current Price* Command Payload

Octets	1
Data Type	uint8
Field Name	Command Options

10.2.2.3.1.1.1 Payload Details

The Command Options Field: The command options field is 8 Bits in length and is formatted as a bit field as shown in Figure 10-3.

Figure 10-3. *Get Current Price* Command Options Field

Bits	0	1 to 7
Field Name	Requestor Rx On When Idle	Reserved

The Requestor Rx On When Idle Sub-field: The Requestor Rx On When Idle sub-field has a value of 1 if the requestor's receiver may be, for all practical purposes, enabled when the device is not actively transmitting, thereby making it very likely that regular broadcasts of pricing information will be received by this device, and 0 otherwise.

A device that publishes price information may use the value of this bit, as received from requestors in its neighborhood, to determine publishing policy. For example, if a device makes a request for current pricing information and the requestor Rx on when idle sub-field of the GetCurrentPrice command payload has a value of 1 (indicating that the device will be likely to receive regular price messages), then the receiving device may store information about the requestor and use it in future publishing operations.

10.2.2.3.1.2 Effect on Receipt

On receipt of this command, the device shall send a Publish Price command (sub-clause 10.2.2.4.1) for the currently scheduled time. Please note: The PublishPrice command is sent out on the network from which the GetCurrentPrice command was received (either the Inter-Pan or SE network). Example: If the GetCurrentPrice command is received on the Inter-Pan network, the ESI shall respond on the Inter-Pan. If the GetCurrentPrice command is received on the SE Network, the ESI shall respond to the device requesting the pricing information.

10.2.2.3.2 Get Scheduled Prices Command

This command initiates a Publish Price command (see sub-clause 10.2.2.4.1) for available price events. A server device shall be capable of storing five price events at a minimum.

10.2.2.3.2.1 Payload Details

The Get Scheduled Prices command payload shall be formatted as illustrated in Figure 10-4.

Figure 10-4. Format of the *Get Scheduled Prices* Command Payload

Octets	4	1
Data Type	UTC	uint8
Field Name	Start Time (M)	Number of Events (M)

Start Time (mandatory): UTC Timestamp representing the minimum ending time for any scheduled or currently active pricing events to be resent. If a command has a Start Time of 0x00000000, replace that Start Time with the current time stamp.

Number of Events (mandatory): Represents the maximum number of events to be sent. A value of 0 would indicate all available events are to be returned. Example: Number of Events = 1 would return the first event with an EndTime greater than or equal to the value of Start Time field in the Get Scheduled Prices command. (EndTime would be StartTime plus Duration of the event listed in the device's event table).

10.2.2.3.2.2 When Generated

This command is generated when the client device wishes to verify the available Price Events or after a loss of power/reset occurs and the client device needs to recover currently active, scheduled, or expired Price Events.

A ZCL Default Response with status NOT_FOUND shall be returned if there are no events available.

10.2.2.3.2.3 Effect on Receipt

On receipt of this command, the device shall send a Publish Price command (see sub-clause 10.2.2.4.1) for all currently scheduled price events. Please note: The Publish Price command is sent out on the network from which the GetScheduledPrices command was received (either the Inter-Pan or SE network). Example: If the GetScheduledPrices command is received on the Inter-Pan network, the ESI shall respond on the Inter-Pan. If the GetScheduledPrices command is received on the SE Network, the ESI shall respond to the device requesting the pricing information.

10.2.2.3.3 Price Acknowledgement Command

The Price Acknowledgement command described in Figure 10-5 provides the ability to acknowledge a previously sent Publish Price command. It is mandatory for 1.1 and later devices. For SE 1.0 devices, the command is optional.

10.2.2.3.3.1 Payload Format

Figure 10-5. Format of the *Price Acknowledgement* Command Payload

Octets	4	4	4	1
Data Type	uint32	uint32	UTC	map8
Field Name	Provider ID (M)	Issuer Event ID (M)	Price Ack Time (M)	Control (M)

10.2.2.3.3.1.1 Payload Details

Provider ID (mandatory): An unsigned 32-bit field containing a unique identifier for the commodity provider.

Issuer Event ID (mandatory): Unique identifier generated by the commodity provider.

Price Ack Time (mandatory): Time price acknowledgement generated.

Control (mandatory): Identifies the Price Control or Block Period Control options for the event. The values for this field are described in Figure 10-9 and Figure 10-10.

10.2.2.3.3.2 When Generated

This command is generated on receipt of a Publish Price command when the Price Control field of that Publish Price command indicates that a Price Acknowledgement is required (see sub-clause 10.2.2.4.1 for further details).

10.2.2.3.4 Get Block Period(s) Command

This command initiates a Publish Block Period command (see sub-clause 10.2.2.4.2) for the currently scheduled block periods. A server device shall be capable of storing at least two commands, the current period and a period to be activated in the near future.

Note: The Get Block Period(s) command in this revision of this specification is provisional and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

10.2.2.3.4.1 Payload Format

Figure 10-6. Format of the *Get Block Period(s)* Command Payload

Octets	4	1
Data Type	UTC	uint8
Field Name	Start Time (M)	Number of Events (M)

10.2.2.3.4.1.1 Payload Details

Start Time (mandatory): UTC Timestamp representing the minimum ending time for any scheduled or currently block period events to be resent. If a command has a Start Time of 0x00000000, replace that Start Time with the current time stamp.

Number of Events (mandatory): An 8 bit integer which indicates the maximum number of Publish Block Period commands that can be sent. Example: Number of Events = 1 would return the first event with an EndTime greater than or equal to the value of Start Time field in the GetBlockPeriod(s) command. (EndTime would be StartTime plus Duration of the event listed in the device's event table). Number of Events = 0 would return all available Publish Block Periods, starting with the current block in progress.

10.2.2.3.4.2 When Generated

This command is generated when the client device wishes to verify the available Block Period events or after a loss of power/reset occurs and the client device needs to recover currently active or scheduled Block Periods.

A ZCL Default response with status NOT_FOUND shall be returned if there are no events available.

10.2.2.3.4.3 Effect on Receipt

On receipt of this command, the device shall send a Publish Block Period command (sub-clause 10.2.2.4.2) for all currently scheduled periods, up to the maximum number of commands specified.

10.2.2.3.5 GetConversionFactor Command

This command initiates a PublishConversionFactor command for the scheduled conversion factor updates. A server device shall be capable of storing at least two instances, the current and next instance to be activated in the near future (if available).

Note: The GetConversionFactor command in this revision of this specification is provisional and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

10.2.2.3.5.1 Payload Format

Figure 10-7. Format of the *GetConversionFactor* Command Payload

Octets	4	4
Data Type	UTC	uint8
Field Name	Start Time	Number of Events

10.2.2.3.5.2 Payload Details

Start Time (mandatory): UTC Timestamp to select active and scheduled events to be returned by the corresponding PublishConversionFactor command. If command has a Start Time of 0x00000000, replace that Start Time with the current time stamp.

Number of Events (mandatory): An 8-bit integer which represents the maximum number of PublishConversionFactor commands to be sent. A value of 0 would indicate all available PublishConversionFactor commands shall be returned. The first returned PublishConversionFactor command shall be the instance which is active or becomes active at the stated Start Time. If more than one instance is requested, the active and scheduled instances shall be sent with ascending ordered StartTime.

10.2.2.3.6 GetCalorificValue Command

This command initiates a PublishCalorificValue command for the scheduled calorific value updates. A server device shall be capable of storing at least two instances, the current and next instance to be activated in the near future (if available).

Note: The GetCalorificValue command in this revision of this specification is provisional and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

10.2.2.3.6.1 Payload Format

Figure 10-8. Format of the *GetCalorificValue* Command Payload

Octets	4	1
Data Type	UTC	uint8
Field Name	Start Time	Number of Events

10.2.2.3.6.2 Payload Details

Start Time (mandatory): UTC Timestamp to select active and scheduled events to be returned by the corresponding PublishCalorificValue command. If the command has a Start Time of 0x00000000, replace that Start Time with the current time stamp.

Number of Events (mandatory): An 8-bit integer which represents the maximum number of PublishCalorificValue commands to be sent. A value of 0 would indicate all available PublishCalorificValue commands shall be returned. The first returned PublishCalorificValue command shall be the instance which is active at the stated Start Time. If more than one instance is requested, the active and scheduled instances shall be sent with ascending ordered Start Time.

10.2.2.4 Commands Generated

The server side of the Price cluster is capable of generating the commands listed in Table 10-11.

Table 10-11. Generated Command IDs for the Price Cluster

Command Identifier Field Value	Description	M/O
0x00	*Publish Price*	M
0x01	*Publish Block Period*	O
0x02	*Publish Conversion Factor*	O
0x03	*Publish Calorific Value*	O

10.2.2.4.1 Publish Price Command

The Publish Price command is generated in response to receiving a Get Current Price command (see sub-clause 10.2.2.3.1), in response to a Get Scheduled Prices command (see sub-clause 10.2.2.3.2), and when an update to the pricing information is available from the commodity provider, either before or when a TOU price becomes active . Additionally the Publish Price Command is generated as specified in sub-clause 10.2.4.3 when Block Pricing is in effect.

When a Get Current Price or Get Scheduled Prices command is received over a Smart Energy network, the Publish Price command should be sent unicast to the requester. In the case of an update to the pricing information from the commodity provider, the Publish Price command should be unicast to all individually registered devices implementing the Price Cluster on the Smart Energy network. When responding to a request via the Inter- PAN SAP, the Publish Price command should be broadcast to the PAN of the requester after a random delay between 0 and 0.5 seconds, to avoid a potential broadcast storm of packets.

Devices capable of receiving this command must be capable of storing and supporting at least two pricing information instances, the current active price and the next price. By supporting at least two pricing information instances, receiving devices will allow the Publish Price command generator to publish the next pricing information during the current pricing period.

Nested and overlapping Publish Price commands are not allowed. The current active price will be replaced if new price information is received by the ESI. In the case of overlapping events, the event with the newer Issuer Event ID takes priority over all nested and overlapping events. All existing events that overlap, even partially, should be removed. The only exception to this is that if an event with a newer Issuer Event ID overlaps with the end of the current active price but is not yet active, the active price is not deleted but its duration is modified to 0xFFFF (until changed) so that the active price ends when the new event begins.

10.2.2.4.1.1 Payload Format

The PublishPrice command payload shall be formatted as illustrated in Figure 10-9.

Figure 10-9. Format of the *Publish Price* Command Payload

Octets	4	1-13	4	4	1	2	1
Data Type	uint32	octstr	uint32	UTC	enum8	uint16	map8
Field Name	Provider ID (M)	Rate Label (M)	Issuer Event ID (M)	Current Time (M)	Unit of Measure (M)	Currency (M)	Price Trailing Digit & Price Tier (M)

Octets	1	4	2	4	1	4	1
Data Type	map8	UTC	uint16	uint32	uint8	uint32	uint8
Field Name	Number of Price Tiers & Register Tier (M)	Start Time (M)	Duration In Minutes (M)	Price (M)	Price Ratio (O)	Generation Price (O)	Generation Price Ratio (O)

Octets	4	1	1	1[b]	1[c]
Data Type	uint32	enum8	map8	8 bit integer	map8
Field Name	Alternate Cost Delivered (O)	Alternate Cost Unit (O)	Alternate Cost Trailing Digit(O)	Number of Block Thresholds (O)	Price Control (O)

Note: M = Mandatory field, O = Optional field. **All fields must be present in the payload.** Optional fields will be marked with specific values to indicate they are not being used.

Provider ID (mandatory): An unsigned 32-bit field containing a unique identifier for the commodity provider. This field is thought to be useful in deregulated markets where multiple commodity providers may be available.

Rate Label (mandatory): A ZCL Octet String field capable of storing a 12 character string (the first octet indicates length) containing commodity provider- specific information regarding the current billing rate. The String shall be encoded in the UTF-8 format. This field is thought to be useful when a commodity provider may have multiple pricing plans.

Issuer Event ID (mandatory): Unique identifier generated by the commodity provider. When new pricing information is provided that replaces older pricing information for the same time period, this field allows devices to determine which information is newer. It is expected that the value contained in this field is a unique number managed by upstream servers or a UTC based time stamp (UTC data type) identifying when the Publish Price command was issued. Thus, newer pricing information will have a value in the Issuer Event ID field that is larger than older pricing information.

Current Time (mandatory): A UTC field containing the current time as determined by the device. This field is thought to be useful to provide an extra value-added feature for the broadcast price signals.

Unit of Measure (mandatory): An 8-bit enumeration field identifying the commodity as well as its base unit of measure. The enumeration used for this field shall match one of the UnitOfMeasure values using a pure binary format as defined in the Metering cluster (see sub-clause 0).

Currency (mandatory): An unsigned 16-bit field containing identifying information concerning the local unit of currency used in the price field. This field is thought to be useful for displaying the appropriate symbol for a currency (i.e.: $).

The value of the currency field should match the values defined by ISO 4217.

Price Trailing Digit and Price Tier (mandatory): An 8-bit field used to determine where the decimal point is located in the price field and to indicate the current pricing tier as chosen by the commodity provider. The most significant nibble is the Trailing Digit sub-field which indicates the number of digits to the right of the decimal point. The least significant nibble is an enumerated field containing the current Price Tier. Valid values for the Price Tier sub-field are from 1 to 15 reflecting the least expensive tier (1) to the most expensive tier (15). A value of zero indicates no price tier is in use. This sub-field also references the associated TiernPriceLabel attribute assigned to the Price Tier. Table 10-12 depicts the assignments.

Note: Values for Price Tier listed above 0x06 in this revision of this specification are provisional and not certifiable. This number of fields may change before reaching certifiable status in a future revision of this specification.

Table 10-12. Price Tier Sub-field Enumerations

Enumerated Value	Price Tier
0x0	No Tier Related
0x1	Reference *Tier1PriceLabel*
0x2	Reference *Tier2PriceLabel*
0x3	Reference *Tier3PriceLabel*
0x4	Reference *Tier4PriceLabel*
0x5	Reference *Tier5PriceLabel*
0x6	Reference *Tier6PriceLabel*
0x7	Reference *Tier7PriceLabel*
0x8	Reference *Tier8PriceLabel*
0x9	Reference *Tier9PriceLabel*
0xA	Reference *Tier10PriceLabel*
0xB	Reference *Tier11PriceLabel*
0xC	Reference *Tier12PriceLabel*
0xD	Reference *Tier13PriceLabel*
0xE	Reference *Tier14PriceLabel*
0xF	Reference *Tier15PriceLabel*

Number of Price Tiers & Register Tier (mandatory): An 8-bit bitmap where the most significant nibble is an enumerated sub-field representing the maximum number of price tiers available, and the least significant nibble is an enumerated sub-field indicating the register tier used with the current Price Tier. Valid values for the Number of Price Tiers sub-field are from 0 to 15 reflecting no tiers in use (0) to fifteen tiers available (15).

The Register Tier values correlates which CurrentTierNSummationDelivered attribute, found in sub-clause 10.4.2.2.2 is accumulating usage information. Both attributes can be used to calculate and display usage and subsequent costs. Register Tier enumerated values are listed in Table 10-13.

Note: Values for Register Tier Sub-field Enumerations listed above 0x06 in this revision of this specification are provisional and not certifiable. This number of fields may change before reaching certifiable status in a future revision of this specification.

Table 10-13. Register Tier Sub-field Enumerations

Enumerated Value	Register Tier
0x0	No Tier Related
0x1	Usage accumulating in *CurrentTier1SummationDelivered* attribute
0x2	Usage accumulating in *CurrentTier2SummationDelivered* attribute
0x3	Usage accumulating in *CurrentTier3SummationDelivered* attribute
0x4	Usage accumulating in *CurrentTier4SummationDelivered* attribute
0x5	Usage accumulating in *CurrentTier5SummationDelivered* attribute
0x6	Usage accumulating in *CurrentTier6SummationDelivered* attribute
0x7	Usage accumulating in *CurrentTier7SummationDelivered* attribute
0x8	Usage accumulating in *CurrentTier8SummationDelivered* attribute
0x9	Usage accumulating in *CurrentTier9SummationDelivered* attribute
0xA	Usage accumulating in *CurrentTier10SummationDelivered* attribute
0xB	Usage accumulating in *CurrentTier11SummationDelivered* attribute
0xC	Usage accumulating in *CurrentTier12SummationDelivered* attribute
0xD	Usage accumulating in *CurrentTier13SummationDelivered* attribute
0xE	Usage accumulating in *CurrentTier14SummationDelivered* attribute
0xF	Usage accumulating in *CurrentTier15SummationDelivered* attribute

Start Time (mandatory): A UTC field to denote the time at which the price signal becomes valid. A Start Time of 0x00000000 is a special time denoting “now.”

If the device would send a price with a Start Time of now, adjust the Duration In Minutes field to correspond to the remainder of the price.

Duration In Minutes (mandatory): An unsigned 16-bit field used to denote the amount of time in minutes after the Start Time during which the price signal is valid. Maximum value means “until changed”. If Block Charging only is in use (see sub-clause 10.2.4.3 for further details), the Duration in Minutes field of the Publish Price command shall be set to 0xFFFF indicating the price is valid “until changed”.

Price (mandatory): An unsigned 32-bit field containing the price of the commodity measured in base unit of Currency per Unit of Measure with the decimal point located as indicated by the Price Trailing Digit field when the commodity is delivered to the premises.

Price Ratio (optional): An unsigned 8-bit field that gives the ratio of the price denoted in the Price field to the "normal" price chosen by the commodity provider. This field is thought to be useful in situations where client devices may simply be interested in pricing levels or ratios. The value in this field should be scaled by a factor of 0.1, giving a range of ratios from 0.1 to 25.4. A value of 0xFF indicates the field is not used and 0x00 is an invalid value.

Generation Price (optional): An unsigned 32-bit field containing the price of the commodity measured in base unit of Currency per Unit of Measure with the decimal point located as indicated by the Price Trailing Digit field when the commodity is received from the premises. An example use of this field is in energy markets where the price of electricity from the grid is different than the price of electricity placed on the grid. A value of 0xFFFFFFFF indicates the field is not used.

Generation Price Ratio (optional): An unsigned 8-bit field that gives the ratio of the price denoted in the Generation Price field to the "normal" price chosen by the commodity provider. This field is thought to be useful in situations where client devices may simply be interested in pricing levels or ratios. The value in this field should be scaled by a factor of 0.1, giving a range of ratios from 0.1 to 25.4. A value of 0xFF indicates the field is not used and 0x00 is an invalid value.

Alternate Cost Delivered (optional): An unsigned 32 Integer field that provides a mechanism to describe an alternative measure of the cost of the energy consumed. An example of an Alternate Cost might be the emissions of CO_2 for each kWh of electricity consumed providing a measure of the environmental cost. Another example is the emissions of CO_2 for each cubic meter of gas consumed (for gas metering). A different value for each price tier may be provided which can be used to reflect the different mix of generation that is associated with different TOU rates. A value of 0xFFFFFFFF indicates the field is not used.

Alternate Cost Unit (optional): An 8-bit enumeration identifying the unit (as specified in Table 10-14) for the Alternate Cost Delivered field. A value of 0xFF indicates the field is not used.

Table 10-14. Alternate Cost Unit Enumerations

Values	Description
0x01	Kg of CO_2 per unit of measure

Alternate Cost Trailing Digit (optional): An 8-bit bitmap field used to determine where the decimal point is located in the alternate cost field. The most significant nibble indicates the number of digits to the right of the decimal point. The least significant nibble is reserved. A value of 0xFF indicates the field is not used.

Number of Block Thresholds (optional): An 8-bit integer which indicates the number of block thresholds available. Valid values are from 0 to 15 reflecting no blocks in use (0) to 15 block thresholds available (15). A value of 0xFF indicates field not used. Any value between 1 and 15 indicates that Block Pricing shall be used, see sub-clause 10.2.4.3 for further details.

Price Control (optional): Identifies additional control options for the price event. A value of 0x00 indicates field not used. Note that for Smart Energy 1.1 and later devices, the Price Acknowledgement command is mandatory, but for SE 1.0 devices, it was optional, so the sender of the Publish Price command should not rely on receiving a Price Acknowledgment command even if the Price Acknowledgement bit in the Price Control Field is set.

The BitMap for this field is described in Table 10-15.

Table 10-15. Price Control Field BitMap

Bit	Description
0	1=Price Acknowledgement required 0=Price Acknowledgement not required

10.2.2.4.1.2 Effect on Receipt

On receipt of this command, the device is informed of a price event for the specific provider, commodity, and currency indicated.

Should the device choose to change behavior based on the price event, the change of behavior should occur after a random delay between 0 and 5 minutes, to avoid potential spikes that could occur as a result of coordinated behavior changes. Likewise, should a device choose to change behavior based on the expiration of the price event, the change in behavior should occur after a random delay between 0 and 5 minutes.,

10.2.2.4.2 Publish Block Period Command

The Publish Block Period command is generated in response to receiving a Get Block Period(s) command (see sub-clause 10.2.2.3.4) or when an update to the block tariff schedule is available from the commodity provider. When the Get Block Period(s) command is received over the Smart Energy network, the Publish Block Period command(s) should be sent unicast to the requestor. In the case of an update to the block tariff schedule from the commodity provider, the Publish Block Period command should be unicast to all individually registered devices implementing the Price Cluster on the Smart Energy network.

Devices capable of receiving this command must be capable of storing and supporting two block periods, the current active block and the next block. By supporting two block periods, receiving devices will allow the Publish Block Period command generator to publish the next block information during the current block period.

Note: The Publish Block Period command in this revision of this specification is provisional and not certifiable. This feature may change before reaching certifiable status in a future revision of this specification.

10.2.2.4.2.1 Payload Format

Figure 10-10. Format of the *Publish Block Period* Command Payload

Octets	4	4	4	3	1	1
Data Type	uint32	uint32	UTC	uint24	map8	map8
Field Name	Provider ID (M)	Issuer Event ID (M)	Block Period Start Time (M)	Block Period Duration In Minutes (M)	Number of Price Tiers & Number of Block Thresholds(M)	Block Period Control (O)

Note: M = Mandatory field, O = Optional field. **All fields shall be present in the payload**. Optional fields will be marked with specific values to indicate they are not being used.

Provider ID (mandatory): An unsigned 32-bit field containing a unique identifier for the commodity provider. This field is thought to be useful in deregulated markets where multiple commodity providers may be available.

Issuer Event ID (mandatory): Unique identifier generated by the commodity provider. When new block period information is provided that replaces older information for the same period, this field allows devices to determine which information is newer. It is expected that the value contained in this field is a unique number managed by upstream servers or a UTC based time stamp (UTC data type) identifying when the Publish Block

Period command was issued. Thus, newer block period information will have a value in the Issuer Event ID field that is larger than older block information.

Block Period Start Time (mandatory): A UTC field to denote the time at which the block tariff period starts. A start time of 0x00000000 is a special time denoting “now”. If the device would send an event with a Start Time of now, adjust the Duration In Minutes field to correspond to the remainder of the event.

Block Period Duration In Minutes (mandatory): An unsigned 24-bit field to denote the block tariff period in minutes. Maximum value (0xFFFFFF) means 'until changed'.

Number of Price Tiers and Number of Block Thresholds (mandatory): An 8-bit bitmap where the most significant nibble is an enumerated sub-field representing the maximum number of price tiers available, and the least significant nibble is an enumerated sub-field indicating the number of block thresholds available. Valid values for the Number of Price Tiers sub-field are from 0 to 15 reflecting no tiers in use (0) to fifteen tiers available (15). Valid values for the Number of Block Thresholds sub-field are from 0 to 15 reflecting no blocks in use (0) to 15 block thresholds available (15).

Block Period Control (optional): Identifies additional control options for the block period event. A value of 0x00 indicates field not used.

The BitMap for this field is described in Table 10-16.

Table 10-16. Block Period Control Field BitMap

Bit	Description
0	1=Price Acknowledgement required 0=Price Acknowledgement not required
1	1=Repeating Block 0=Non Repeating Block

Repeating Block: Indicates whether a block period repeats on expiry.

10.2.2.4.3 PublishConversionFactor Command

The PublishConversionFactor command is sent in response to a GetConversionFactor command or if a new conversion factor is available.

Clients shall be capable of storing at least two instances of the Calorific Value, the currently active one and the next one.

Note: The PublishConversionFactor command in this revision of this specification is provisional and not certifiable. This feature may change before reaching certifiable status in a future revision of this specification.

10.2.2.4.3.1 Payload Format

Figure 10-11. Format of the *PublishConversionFactor* Command Payload

Octets	4	4	4	1
Data Type	uint32	UTC	uint32	map8
Field Name	Issuer Event ID (M)	Start Time (M)	Conversion Factor (M)	Conversion Factor Trailing Digit (M)

10.2.2.4.3.2 Payload Details

Issuer Event ID (mandatory): Unique identifier generated by the commodity provider.

Start Time (mandatory): A UTC field to denote the time at which the value becomes valid. The value remains valid until replaced by a newer one.

Conversion Factor (mandatory): See Price Cluster Commodity attributes (see sub-clause 10.2.2.2.4.3).

Conversion Factor Trailing Digit (mandatory): See Price Cluster Commodity attributes (see sub-clause 10.2.2.2.4.4).

10.2.2.4.4 PublishCalorificValue Command

The PublishCalorificValue command is sent in response to a GetCalorificValue command or if a new calorific value is available. Clients shall be capable of storing at least two instances of the Calorific Value, the currently active one and the next one.

Note: The PublishCalorificValue command in this revision of this specification is provisional and not certifiable. This feature may change before reaching certifiable status in a future revision of this specification.

10.2.2.4.4.1 Payload Format

Figure 10-12. Format of the *PublishCalorificValue* Command Payload

Octets	4	4	4	1	1
Data Type	uint32	UTC	uint32	enum8	map8
Field Name	Issuer Event ID (M)	Start Time (M)	Calorific Value (M)	Calorific Value Unit (M)	Calorific Value Trailing Digit (M)

10.2.2.4.4.2 Payload Details

Issuer Event ID (mandatory): Unique identifier generated by the commodity provider.

Start Time (mandatory): A UTC field to denote the time at which the value becomes valid. The value remains valid until replaced by a newer one.

Calorific Value (mandatory): See Price Cluster Commodity attributes (see sub- clause 10.2.2.2.4.5).

Calorific Value Unit (mandatory): See Price Cluster Commodity attributes (see sub-clause 10.2.2.2.4.6).

Calorific Value Trailing Digit (mandatory): See Price Cluster Commodity attributes (see sub-clause 10.2.2.2.4.7).

10.2.3 Client

10.2.3.1 Dependencies

Events carried using this cluster include a timestamp with the assumption that target devices maintain a real time clock. Devices can acquire and synchronize their internal clocks via the ZCL Time server.

If a device does not support a real time clock it is assumed that the device will interpret and utilize the “Start Now” 0x00000000 value within the Time field.

Anonymous Inter-PAN transmission mechanism.

Note: The Price Client Cluster Attributes in this revision of this specification are provisional and not certifiable. These features may change before reaching certifiable status in a future revision of this specification.

10.2.3.2 Attributes

Table 10-17. Price Client Cluster Attributes

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*PriceIncreaseRandomizeMinutes*	uint8	0x00 to 0x3C	RW	0x05	O
0x0001	*PriceDecreaseRandomizeMinutes*	uint8	0x00 to 0x3C	RW	0x0F	O
0x0002	*CommodityType*	enum8	0x00 to 0xFF	R	-	O

10.2.3.2.1 PriceIncreaseRandomizeMinutes Attribute

The PriceIncreaseRandomizeMinutes attribute represents the maximum amount of time to be used when randomizing the response to a price increase. Note that although the granularity of the attribute is in minutes, it is recommended the granularity of the randomization used within a responding device be in seconds or smaller. If a device responds to a price increase it must choose a random amount of time, in seconds or smaller, between 0 and PriceIncreaseRandomizeMinutes minutes. The device must implement that random amount of time before or after the price change. How and if a device will respond to a price increase is up to the manufacturer. Whether to respond before or after the price increase is also up to the manufacturer.

As an example, a water heater with a PriceIncreaseRandomizeMinutes set to 6 could choose to lower its set point 315 seconds (but not more than 360 seconds) before the price increases.

The valid range for this attribute is 0x00 to 0x3C.

If PriceIncreaseRandomizeMinutes or PriceDecreaseRandomizeMinutes attributes are not supported by the client, then it should use the default values for the attributes as specified in the Price Client Cluster Attribute table.

10.2.3.2.2 PriceDecreaseRandomizeMinutes Attribute

The PriceDecreaseRandomizeMinutes attribute represents the maximum number of minutes to be used when randomizing the response to a price decrease. Note that although the granularity of the attribute is in minutes, it is recommended the granularity of the randomization used within a responding device be in seconds or smaller. If a device responds to a price decrease it must choose a random amount of time, in seconds or smaller, between 0 and PriceDecreaseRandomizeMinutes minutes and implement that random amount of time before or after the price change. How and if a device will respond to a price decrease is up to the manufacturer. Whether to respond before or after the price increase is also up to the manufacturer.

As an example, a dishwasher with a PriceDecreaseRandomizeMinutes set to 15 could choose to start its wash cycle 723 seconds (but not more than 900 seconds) after the price decreases.

The valid range for this attribute is 0x00 to 0x3C.

10.2.3.2.3 CommodityType Attribute

CommodityType provides a label for identifying the type of pricing client present. The attribute is an enumerated value representing the commodity. The defined values are represented by the non-mirrored values (0-127) in the MeteringDeviceType attribute enumerations (refer to Table 10-42).

10.2.3.3 Commands Received

The client receives the cluster-specific response commands detailed in sub-clause 10.2.2.4.

10.2.3.4 Commands Generated

The client generates the cluster-specific commands detailed in sub-clause 10.2.2.3, as required by the application.

10.2.4 Application Guidelines

10.2.4.1 Registering for Commands

Devices should use bind request to register for unsolicited Publish Price, Display Message and Load Control Event commands.

10.2.4.2 Attribute Reporting

Attribute reporting may be used for sending information in the Price Server Cluster Attributes table. The Price Cluster attributes can be polled periodically for updates. Polling should not occur more frequently than recommended in 10.4.3.2. Use of the Report Attribute command without report configuration may be used for unsolicited notification of an attribute value change. Sleepy devices may have to poll.

10.2.4.3 Block Tariffs

Upon reaching the Start Time of a received Publish Price command, a device's behavior will depend on the values of the Number of Block Thresholds and Number of Price Tiers fields. A client device needing to determine if it should use Block Pricing shall send a Get Current Price command to the Price server and check the Number of Block Thresholds in the Publish Price response. Any value between 1 and 15 indicates that Block Pricing shall be used.

The prices for a commodity being delivered to the premises shall be taken from the Block Pricing Information Attribute Set whenever Block Pricing is active.

10.2.4.3.1 TOU Charging Only

Indicated by the Number of Block Thresholds field being set to zero. Charging shall be according to the price fields within the Publish Price command itself.

10.2.4.3.2 Block Charging Only

Indicated by the Number of Price Tiers fields being set to zero while the Number of Block Thresholds is between 0x01 and 0x0F.

A server shall not update the Block Threshold and Block Price attribute sets of an active Block Period. Updates to these attribute sets can only be done by creating a new Block Period. The server may create a new active Block Period by updating either Block Period Start Time (attribute StartOfBlockPeriod) alone or Block Period Duration in Minutes (attribute BlockPeriodDuration) followed by Block Period Start Time (attribute StartOfBlockPeriod) along with updating other attributes as desired.

When a server transmits a Publish Price command it shall additionally fill fields necessary to support backwards compatibility with clients that may not support Block Charging. The Price field shall be set according to the Block Price Information Attribute Set. The Duration in Minutes field shall be set to 0xFFFF indicating the price is valid "until changed".

A server shall additionally transmit a Publish Price command to clients under the following conditions:

1. At the start of a Block Period
2. When it is notified that a Block Threshold has been crossed

3. When *Block Period Start Time* or *Block Period Duration in Minutes* have changed to indicate a new active block period

A client may cache attributes from the Block Threshold, Block Period, Block Price, and Billing Period attribute sets. Cached attributes are valid only during the active Block Period when received. Upon reaching Block Period Start Time or detecting a new active Block Period, the client should retrieve updated values for cached attributes.

A client shall check for a new active Block Period on receipt of an asynchronous Publish Price command (i.e. not required on a Publish Price command in response to Get Current Price) by checking Block Period Start Time and Block Period Duration in Minutes for update. Additionally, it shall infrequently (e.g. once an hour) query the StartOfBlockPeriod and BlockPeriodDuration attributes to verify that the Block Period has not ended early.

10.2.4.3.3 Block/TOU Combination Charging

Note: The following application guidelines that pertain to Block/TOU Combination Charging in this revision of this specification are provisional and not certifiable. This text may change before reaching certifiable status in a future revision of this specification.

The Number of Block Thresholds and Number of Price Tiers fields will both be set to non-zero values, indicating the number of blocks and number of tiers respectively being used. The start of a Block period shall be indicated by the value of the Block Period Start Time field within a Publish Block Period command. Upon reaching the Block Period Start Time, the attributes for the required number of Block Thresholds, together with the Block Prices for all required blocks for the selected tier should be fetched from the server. The Block Period Duration in Minutes field shall indicate the length of the block period.

A Publish Price command will be received for the start of each new TOU period during a block period. At this point the attributes for the Block Prices for all required blocks for the newly activated tier should be fetched from the server.

10.2.4.3.4 Application Guidelines for Block Pricing Under Specific Events

HAN device not communicating with meter for extended period of time:

In this situation, when the HAN device reconnects with the meter, it will need to read the Block Information Set to calculate the correct cost for the given period. This is done by applying the prices for each block/tier combination to the consumption information for each block/tier combination. If a block period has passed while the HAN device was not communicating with the meter, then the prior period consumption information will not be known and the prior period cost cannot be calculated by the HAN device.

Meter installation or swap-out:

The new meter will need to be configured with the appropriate block thresholds, pricing, and block duration by the utility. If this does not occur precisely at the start of that customer's billing period, the utility will need to (a) pro-rate these amounts over the remaining billing period duration and (b) decide how to handle the initial portion of the period. Any information from the initial part of the billing period will be lost when the new meter is installed. As such, HAN devices may not display accurate information for this billing period and utilities should advise customers of this situation. As a typical meter lifetime is expected to be in the range of 10 to 20 years, this event is expected to be rare.

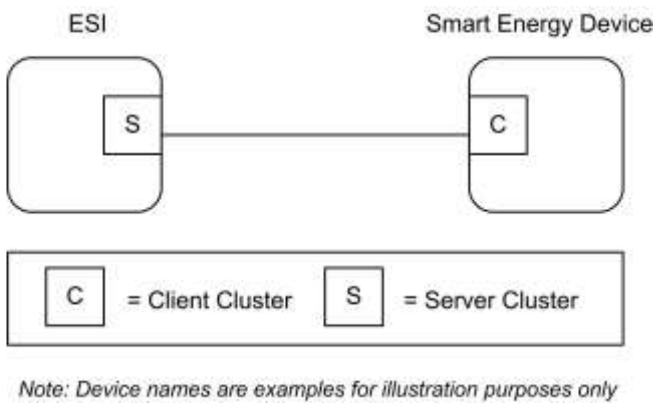
10.3 Demand Response and Load Control

10.3.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface to the functionality of Smart Energy Demand Response and Load Control. Devices targeted by this cluster include thermostats and devices that support load control.

Figure 10-13. Demand Response/Load Control Cluster Client Server Example


ESI
Smart Energy Device
S
C
C
= Client Cluster
S
= Server Cluster
Note: Device names are examples for illustration purposes only

Please note the ESI is defined as the Server due to its role in acting as the proxy for upstream demand response/load control management systems and subsequent data stores.

10.3.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

10.3.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	DRLC	Type 1 (client to server)

10.3.1.3 Cluster Identifiers

Identifier	Name
0x0701	Demand Response and Load Control

10.3.2 Server

By default the ESI will be labeled as the Server side in the cluster descriptions, being able to initiate load control commands to other devices in the network.

10.3.2.1 Dependencies

A server device shall be capable of storing at least two load control events.

Events carried using this cluster include a timestamp with the assumption that target devices maintain a real-time clock. Devices can acquire and synchronize their internal clocks with the ESI as described in sub-clause 3.12.

If a device does not support a real-time clock, it is assumed the device will ignore all values within the Time field except the “Start Now” value.

Additionally, for devices without a real-time clock, it is assumed those devices will utilize a method (i.e. ticks, countdowns, etc.) to approximate the correct duration period.

10.3.2.2 Attributes

There are no attributes for the Demand Response and Load Control Cluster server.

10.3.2.3 Commands Generated

The command IDs generated by the Demand Response and Load Control cluster server are listed in Table 10-18.

Table 10-18. Command IDs for the Demand Response and Load Control Server

Command Identifier	Description	M/O
0x00	Load Control Event	M
0x01	Cancel Load Control Event	M
0x02	Cancel All Load Control Events	M

10.3.2.3.1 Load Control Event Command

10.3.2.3.1.1 Payload Format

The Load Control Event command payload shall be formatted as illustrated in Figure 10-14.

Figure 10-14. Format of the *Load Control Event* Command Payload

Octets	4	2	1	4	2	1	1
Data Type	uint32	map16	uint8	UTC	uint16	uint8	uint8
Field Name	Issuer Event ID (M)	Device Class (M)	Utility Enrollment Group (M)	Start Time (M)	Duration in Minutes (M)	Criticality Level (M)	Cooling Temperature Offset (O)

Octets	1	2	2	1	1	1
Data Type	uint8	int16	int16	int8	uint8	map8
Field Name	Heating Temperature Offset (O)	Cooling Temperature Set Point (O)	Heating Temperature Set Point (O)	Average Load Adjustment Percentage (O)	Duty Cycle (O)	Event Control (M)

Note: M = Mandatory field, O = Optional field. **All fields must be present in the payload.** Optional fields will be marked with specific values to indicate they are not being used.

10.3.2.3.1.1.1 Payload Details

Issuer Event ID (mandatory): Unique identifier generated by the Energy provider. The value of this field allows matching of Event reports with a specific Demand Response and Load Control event. The expected value contained in this field shall be a unique number managed by upstream systems or a UTC based time stamp (UTC data type) identifying when the Load Control Event was issued.

Device Class (mandatory): Bit encoded field representing the Device Class to apply the current Load Control Event. Each bit, if set individually or in combination, indicates the class device(s) needing to participate in the event. (Note that the participating device may be different than the controlling device. For instance, a thermostat may act on behalf of an HVAC compressor or furnace and/or Strip Heat/Baseboard Heater and should take action on their behalf, as the thermostat itself is not subject to load shed but controls devices that are subject to load shed.) The encoding of this field is in Table 10-19.

Table 10-19. Device Class Field BitMap/Encoding

Bit	**Description**
0	HVAC Compressor or Furnace
1	Strip Heaters/Baseboard Heaters
2	Water Heater
3	Pool Pump/Spa/Jacuzzi
4	Smart Appliances
5	Irrigation Pump
6	Managed Commercial & Industrial (C&I) loads
7	Simple misc. (Residential On/Off) loads
8	Exterior Lighting
9	Interior Lighting
10	Electric Vehicle
11	Generation Systems

Device manufacturers shall recognize the Device Class or set of Devices Classes that corresponds to its functionality. For example, a thermostat (PCT) may react when Bit 0 is set since it controls the HVAC and/or furnace. Another example is a device that acts like an EMS where it controls exterior lights, interior lights, and simple misc. load control devices. In this case the EMS would react when Bits 7, 8, or 9 are set individually or in combination.

Utility Enrollment Group (mandatory): The Utility Enrollment Group field can be used in conjunction with the Device Class bits. It provides a mechanism to direct Load Control Events to groups of Devices. Example, by assigning two different groups relating to either Demand Response programs or geographic areas, Load Control Events can be further directed for a sub-set of Device Classes (i.e. Device Class Bit 0 and Utility Enrollment Group #1 vs. Device Class Bit0 and Utility Enrollment Group #2). 0x00 addresses all groups, and values 0x01 to 0xFF address individual groups that match. Please refer to sub-clause 10.3.3.2.1 for further details.

If the Device Class and/or Utility Enrollment Group fields don't apply to your End Device, the Load Control Event command shall be ignored by either dropping the message and not replying at all or by sending back a Default Response message with a SUCCESS status code.

Start Time (mandatory): UTC Timestamp representing when the event is scheduled to start. A start time of 0x00000000 is a special time denoting "now." If the device would send an event with a Start Time of now, adjust the Duration In Minutes field to correspond to the remainder of the event.

Duration In Minutes (mandatory): Duration of this event in number of minutes. Maximum value is 1440 (one day).

Criticality Level (mandatory): This field defines the level of criticality of this event. The action taken by load control devices for an event can be solely based on this value, or combination with other Load Control Event fields supported by this device. For example, additional fields such as Average Load Adjustment Percentage, Duty Cycle, Cooling Temperature Offset, Heating Temperature Offset, Cooling Temperature Set Point or Heating Temperature Set Point can be used in combination with the Criticality level. Criticality levels are listed in Table 10-20.

Table 10-20. Criticality Levels

Criticality Level	Level Description	Participation
1	Green	Voluntary
2	1	Voluntary
3	2	Voluntary
4	3	Voluntary
5	4	Voluntary
6	5	Voluntary
7	Emergency	Mandatory
8	Planned Outage	Mandatory
9	Service Disconnect	Mandatory
0x0A to 0x0F	Utility Defined	Utility Defined

The criticality level 0x0 and 0x10 to 0xFF are reserved for future profile changes and not used.

"Green" event, level 0x01, may be used to denote that the energy delivered uses an abnormal amount from non-"green" sources. Participation in this event is voluntary.

The criticality levels 0x02 through 0x06 (Levels 1 through 5) indicate progressively increasing levels of load reduction are being requested by the utility. Participation in these events is voluntary.

The criticality level 0x07 is used to indicate an "Emergency" event. Participation in this event is mandatory, as defined by the utility. The expected response to this event is termination of all non-essential energy use, as defined by the utility. Exceptions to participation in this event type must be managed by the utility.

The criticality level 0x08 is used to indicate a "Planned Outage" event. Participation in this event is mandatory, as defined by the utility. The expected response to this event is termination of delivery of all non-essential energy, as defined by the utility. Exceptions to participation in this event type must be managed by the utility.

The criticality level 0x09 is used to indicate a "Service Disconnect" event. Participation in this event is mandatory, as defined by the utility. The expected response to this event is termination of delivery of all non-essential energy, as defined by the utility. Exceptions to participation in this event type must be managed by the utility.

Levels 0x0A to 0x0F are available for Utility Defined criticality levels.

Cooling Temperature Offset (optional): Requested offset to apply to the normal cooling setpoint at the time of the start of the event in + 0.1 ºC.

Heating Temperature Offset (optional): Requested offset to apply to the normal heating setpoint at the time of the start of the event in + 0.1 ºC.

The Cooling and Heating Temperature Offsets represent a temperature change (Delta Temperature) that will be applied to both the associated heating and cooling set points. The temperature offsets (Delta Temperatures) will be calculated per the Local Temperature in the Thermostat. The calculated temperature will be interpreted as the number of degrees to be added to the cooling set point and subtracted from the heating set point. Sequential demand response events are not cumulative. The Offset shall be applied to the normal setpoint.

Each offset represents the temperature offset (Delta Temperature) in degrees Celsius, as follows: Delta Temperature Offset / 10 = delta temperature in degrees Celsius. Where 0.00°C <= temperature <= 25.4 ºC, corresponding to a Temperature in the range 0x00 to 0x0FE. The maximum resolution this format allowed is 0.1 ºC.

A DeltaTemperature of 0xFF indicates that the temperature offset is not used.

If a temperature offset is sent that causes the heating or cooling temperature set point to exceed the limit boundaries that are programmed into the thermostat, the thermostat should respond by setting the temperature at the limit.

Cooling Temperature Set Point (optional): Requested cooling set point in 0.01 degrees Celsius.

Heating Temperature Set Point (optional): Requested heating set point in 0.01 degrees Celsius.

Cooling and heating temperature set points will be defined and calculated per the LocalTemperature attribute in the Thermostat Cluster (see Chapter 6).

These fields represent the temperature in degrees Celsius, as follows:

Cooling Temperature Set Point / 100 = temperature in degrees Celsius

where -273.15°C <= temperature <= 327.67°C, corresponding to a Cooling and/or Heating Temperature Set Point in the range 0x954d to 0x7fff

The maximum resolution this format allows is 0.01°C.

A Cooling or Heating Temperature Set Point of 0x8000 indicates that the temperature set point is not used.

If a temperature is sent that exceeds the temperature limit boundaries that are programmed into the thermostat, the thermostat should respond by setting the temperature at the limit.

The thermostat shall not use a Cooling or Heating Temperature Set Point that causes the device to use more energy than the normal setting.

When both a Temperature Offset and a Temperature Set Point are provided, the thermostat may use either as defined by the device manufacturer. The thermostat should use the setting that provides the lowest energy consumption.

Average Load Adjustment Percentage (optional): Defines a maximum energy usage limit as a percentage of the client implementations specific average energy usage. The load adjustment percentage is added to 100% creating a percentage limit applied to the client implementations specific average energy usage. A -10% load adjustment percentage will establish an energy usage limit equal to 90% of the client implementations specific average energy usage. Each load adjustment percentage is referenced to the client implementations specific average energy usage. There are no cumulative effects.

The range of this field is -100 to +100 with a resolution of 1 percent. A -100% value equals a total load shed. A 0% value will limit the energy usage to the client implementation's specific average energy usage. A +100% value will limit the energy usage to double the client implementation's specific average energy usage.

A value of 0x80 indicates the field is not used. All other values are reserved for future use.

Duty Cycle (optional): Defines the maximum On state duty cycle as a percentage of time. Example, if the value is 80, the device would be in an "on state" for 80% of the time for the duration of the event. Range of the value is 0 to 100. A value of 0xFF indicates the field is not used. All other values are reserved for future use.

Duty cycle control is a device specific issue and shall be managed by the device manufacturer. It is expected that the duty cycle of the device under control will span the shortest practical time period in accordance with the nature of the device under control and the intent of the request for demand reduction. For typical Device Classes, three minutes for each 10% of duty cycle is recommended. It is expected that the "off state" will precede the "on state".

Event Control (mandatory): Identifies additional control options for the event. The BitMap for this field is described in Table 10-21.

Table 10-21. Event Control Field BitMap

Bit	Description
0	1= Randomize Start time, 0=Randomized Start not Applied
1	1= Randomize End time, 0=Randomized End not Applied

Note: The randomization attribute will be used in combination with two bits to determine if the Event Start and Stop Times are randomized. By default devices will randomize the start and stop of an event. Refer to sub-clause 10.3.3.2.2 and sub-clause 10.3.3.2.3 for the settings of these values.

10.3.2.3.1.1.2 When Generated

This command is generated when the ESI wants to control one or more load control devices, usually as the result of an energy curtailment command from the Smart Energy network.

10.3.2.3.1.1.3 Responses to Load Control Event

The server receives the cluster-specific commands detailed in sub-clause 10.3.3.3.1.

10.3.2.3.2 Cancel Load Control Event Command

10.3.2.3.2.1 Payload Format

The Cancel Load Control Event command payload shall be formatted as illustrated in Figure 10-15.

Figure 10-15. Format of the *Cancel Load Control Event* Payload

Octets	4	2	1	1	4
Data Type	uint32	map16	uint8	map8	UTC
Field Name	Issuer Event ID	Device Class (M)	Utility Enrollment Group (M)	Cancel Control (M)	Effective Time (M)

10.3.2.3.2.1.1 Payload Details

Issuer Event ID (mandatory): Unique identifier generated by the Energy provider. The value of this field allows matching of Event reports with a specific Demand Response and Load Control event. It's expected the value contained in this field is a unique number managed by upstream systems or a UTC based time stamp (UTC data type) identifying when the Load Control Event was issued.

Device Class (mandatory): Bit encoded field representing the Device Class to apply the current Load Control Event. Each bit, if set individually or in combination, indicates the class device(s) needing to participate in the event. (Note that the participating device may be different than the controlling device. For instance, a thermostat may act on behalf of an HVAC compressor or furnace and/or Strip Heat/Baseboard Heater and should take action on their behalf, as the thermostat itself is not subject to load shed but controls devices that are subject to load shed.) The encoding of the Device Class is listed in Table 10-19.

Utility Enrollment Group (mandatory): The Utility Enrollment Group field can be used in conjunction with the Device Class bits. It provides a mechanism to direct Load Control Events to groups of Devices. Example, by assigning two different groups relating to either Demand Response programs or geographic areas, Load Control Events can be further directed for a sub-set of Device Classes (i.e. Device Class Bit 0 and Utility Enrollment Group #1 vs. Device Class Bit0 and Utility Enrollment Group #2). 0x00 addresses all groups, and values 0x01 to 0xFF address individual groups that match. Please refer to sub-clause 10.3.2.3.2.1 for further details.

If the Device Class and/or Utility Enrollment Group fields don't apply to your End Device, the Cancel Load Control Event command is ignored.

Device Class and/or Utility Group fields must be the same for a Cancel Load Control Event command as they were for the command to create the event. Should these fields be different there is no defined behavior for how DRLC servers should maintain their tables for replying to Get Scheduled Events commands.

Cancel Control (mandatory): The encoding of the Cancel Control is listed in Table 10-22.

Table 10-22. Cancel Control

Bit	Description
0	To be used when the Event is currently in process and acted upon as specified by the Effective Time field of the Cancel Load Control Event command. A value of Zero (0) indicates that randomization is overridden and the event should be terminated immediately at the Effective Time. A value of One (1) indicates the event should end using randomization settings in the original event.

Effective Time (mandatory): UTC Timestamp representing when the canceling of the event is scheduled to start. An effective time of 0x00000000 is a special time denoting “now.” If the device would send an event with an Effective Time of now, adjust the Duration In Minutes field to correspond to the remainder of the event.

Note: This field is deprecated; a Cancel Load Control command shall now take immediate effect. A value of 0x00000000 shall be used in all Cancel Load Control commands

10.3.2.3.2.1.2 When Generated

This command is generated when the ESI wants to cancel previously scheduled control of one or more load control devices, usually as the result of an energy curtailment command from the Smart Energy network.

10.3.2.3.2.1.3 Responses to Cancel Load Control Event

The server receives the cluster-specific commands detailed in sub-clause 10.3.3.3.1.

Note: If the Cancel Load Control Event command is received after the event has ended, the device shall reply using the “Report Event Status Command” with an Event Status of “Rejected -Invalid Cancel Command (Undefined Event)”.

10.3.2.3.3 Cancel All Load Control Events Command

10.3.2.3.3.1 Payload Format

The Cancel All Load Control Events command payload shall be formatted as illustrated in Table 10-23.

Table 10-23. Format of the *Cancel All Load Control Events* Command Payload

Octets	1
Data Type	map8
Field Name	Cancel Control

10.3.2.3.3.1.1 Payload Details

Cancel Control: The encoding of the Cancel Control is listed in Table 10-24.

Table 10-24. Cancel All Command Cancel Control Field

Bit	Description
0	To be used when the Event is currently in process and a cancel command is received. A value of Zero (0) indicates that randomization is overridden and the event should be terminated immediately. A value of One (1) indicates the event should end using randomization settings in the original event.

10.3.2.3.3.2 When Generated

This command is generated when the ESI wants to cancel all events for control device(s).

10.3.2.3.3.3 Responses to Cancel All Load Control Events

The server receives the cluster-specific commands detailed in sub-clause 10.3.3.1. The Cancel All Load Control Events command is processed by the device as if individual Cancel Load Control Event commands were received for all of the currently stored events in the device. The device will respond with a “Report Event Status Command” for each individual load control event canceled.

10.3.2.4 Commands Received

The server receives the cluster-specific commands detailed in sub-clause 10.3.3.

10.3.3 Client

This section identifies the attributes and commands provided by Client devices.

10.3.3.1 Dependencies

Devices receiving and acting upon Load Control Event commands must be capable of storing and supporting at least three unique instances of events. As a highly recommended recovery mechanism, when maximum storage of events has been reached and additional Load Control Events are received that are unique (not superseding currently stored events), devices should ignore additional Load Control Events and when storage becomes available, utilize the GetScheduledEvents command to retrieve any previously ignored events.

Events carried using this cluster include a timestamp with the assumption that target devices maintain a real time clock. Devices can acquire and synchronize their internal clocks with the ESI as described in the Time cluster sub-clause 3.12.

Devices MAY ‘drop’ events received before they have received and resolved time (‘dropping’ an event is defined as sending a default response with status code SUCCESS).

If a device does not support a real time clock, it's assumed the device will ignore all values within the Time field except the “Start Now” value.

Additionally, for devices without a real time clock it's assumed those devices will utilize a method (i.e. ticks, countdowns, etc.) to approximate the correct duration period.

10.3.3.2 Client Cluster Attributes

Table 10-25. Demand Response Client Cluster Attributes

Id	Name	Type	Range	Acc	Default	M/O
0x0000	*UtilityEnrollmentGroup*	uint8	0x00 to 0xFF	RW	0x00	M
0x0001	*StartRandomizeMinutes*	uint8	0x00 to 0x3C	RW	0x1E	M
0x0002	*StopRandomizeMinutes*	uint8	0x00 to 0x3C	RW	0x1E	M
0x0003	*DeviceClassValue*	uint16	0x0000 to 0xFFFF	RW	-	M

10.3.3.2.1 *UtilityEnrollmentGroup* Attribute

The UtilityEnrollmentGroup provides a method for utilities to assign devices to groups. In other words, Utility defined groups provide a mechanism to arbitrarily group together different sets of load control or demand response devices for use as part of a larger utility program. The definition of the groups, implied usage, and their assigned values are dictated by the Utilities and subsequently used at their discretion, therefore outside the scope of this specification. The valid range for this attribute is 0x00 to 0xFF, where 0x00 (the default value) indicates the device is a member of all groups and values 0x01 to 0xFF indicates that the device is member of that specified group.

10.3.3.2.2 *StartRandomizationMinutes* Attribute

The StartRandomizedMinutes represents the maximum number of minutes to be used when randomizing the start of an event. As an example, if StartRandomizedMinutes is set for 3 minutes, the device could randomly select 2 minutes (but never greater than the 3 minutes) for this event, causing the start of the event to be delayed by two minutes. The valid range for this attribute is 0x00 to 0x3C where 0x00 indicates start event randomization is not performed.

10.3.3.2.3 *EndRandomizationMinutes* Attribute

The EndRandomizedMinutes represents the maximum number of minutes to be used when randomizing the end of an event. As an example, if EndRandomizedMinutes is set for 3 minutes, the device could randomly select one minute (but never greater than 3 minutes) for this event, causing the end of the event to be delayed by one minute. The valid range for this attribute is 0x00 to 0x3C where 0x00 indicates end event randomization is not performed.

10.3.3.2.4 *DeviceClassValue* Attribute

The DeviceClassValue attribute identifies which bits the device will match in the Device Class fields. Please refer to Table 10-19 for further details. Although the attribute has a RW access property, the device is permitted to refuse to change the DeviceClass by setting the status field of the corresponding write attribute status record to NOT_AUTHORIZED.

Although, for backwards compatibility, the Type cannot be changed, this 16-bit integer should be treated as if it were a 16-bit bitmap.

Device Class and/or Utility Enrollment Group fields are to be used as filters for deciding to accept or ignore a Load Control Event or a Cancel Load Control Event command. There is no requirement for a device to store or remember the Device Class and/or Utility Enrollment Group once the decision to accept the event has been made. A consequence of this is that devices that accept multiple device classes may have an event created for one device class superseded by an event created for another device class.

In-Home Displays should report the device classes that they are interested in. An IHD that wishes to display all possible Load Control Events, even for classes not yet defined, should indicate a device class of 0xFFFF; this will allow DRLC servers to optimize the number of DRLC events they unicast, such that they are only sent to those devices that are interested in them.

10.3.3.3 Commands Generated

The command IDs generated by the Demand Response and Load Control client cluster are listed in Table 10-26.

Table 10-26. Generated Command IDs for the Demand Response and Load Control Client

Command Identifier Field Value	Description	M/O
0x00	*Report Event Status*	M
0x01	*Get Scheduled Events*	M

10.3.3.3.1 *Report Event Status* Command

10.3.3.3.1.1 Payload Format

The Report Event Status command payload shall be formatted as illustrated in Figure 10-16.

Figure 10-16. Format of the *Report Event Status* Command Payload

Octets	4	1	4	1	2	2
Data Type	uint32	uint8	UTC	uint8	uint16	uint16
Field Name	Issuer Event ID (M)	Event Status (M)	Event Status Time (M)	Criticality Level Applied (M)	Cooling Temperature Set Point Applied (O)	Heating Temperature Set Point Applied (O)

Octets	1	1	1	1	42
Data Type	int8	uint8	map8	uint8	opaque
Field Name	Average Load Adjustment Percentage Applied (O)	Duty Cycle Applied (O)	Event Control (M)	Signature Type (M)	Signature (O)

10.3.3.3.1.1.1 Payload Details

Issuer Event ID (mandatory): Unique identifier generated by the Energy provider. The value of this field allows matching of Event reports with a specific Demand Response and Load Control event. It's expected the value contained in this field is a unique number managed by upstream systems or a UTC based time stamp (UTC data type) identifying when the Load Control Event was issued.

Event Status (mandatory): Table 10-27 lists the valid values returned in the Event Status field.

Table 10-27. Event Status Field Values

Value	Description
0x01	Load Control Event command received
0x02	Event started
0x03	Event completed
0x04	User has chosen to “Opt-Out”, user will not participate in this event
0x05	User has chosen to “Opt-In”, user will participate in this event
0x06	The event has been cancelled
0x07	The event has been superseded
0x08	Event partially completed with User “Opt-Out”
0x09	Event partially completed due to User “Opt-In”
0x0A	Event completed, no User participation (Previous “Opt-Out”)
0xF8	Rejected -Invalid Cancel Command (Default)
0xF9	Rejected -Invalid Cancel Command (Invalid Effective Time)
0xFB	Rejected -Event was received after it had expired (Current Time > Start Time + Duration)
0xFD	Rejected -Invalid Cancel Command (Undefined Event)
0xFE	Load Control Event command Rejected

Should a device issue one or more “OptOut” or “OptIn” RES commands during an event that is eventually cancelled, the event shall be recorded as a cancelled event (Status = 0x06) at its effective time.

Should a device issue one or more “OptOut” or “OptIn” RES commands during an event that is not cancelled, the event shall be recorded as partially completed based on the last RES command sent (Status = 0x08 or 0x09).

When a device returns a status of 0xFD (Rejected -Invalid Cancel Command (Undefined Event)), all optional fields should report their “Ignore” values.

When a device receives a duplicate RES command, it should ignore the duplicate commands. Please note: As a recommended best practice, ESI applications should provide a mechanism to assist in filtering duplicate messages received on the WAN.

Event Status Time (mandatory): UTC Timestamp representing when the event status occurred. This field shall not use the value of 0x00000000.

Criticality Level Applied (mandatory): Criticality Level value applied by the device, see the corresponding field in the Load Control Event command for more information.

Cooling Temperature Set Point Applied (optional): Cooling Temperature Set Point value applied by the device, see the corresponding field in the Load Control Event command for more information. The value 0x8000 means that this field has not been used by the end device.

Heating Temperature Set Point Applied (optional): Heating Temperature Set Point value applied by the device, see the corresponding field in the Load Control Event command for more information. The value 0x8000 means that this field has not been used by the end device.

Average Load Adjustment Percentage Applied (optional): Average Load Adjustment Percentage value applied by the device, see the corresponding field in the Load Control Event command for more information. The value 0x80 means that this field has not been used by the end device.

Duty Cycle Applied (optional): Defines the maximum On state duty cycle applied by the device. The value 0xFF means that this field has not been used by the end device. Refer to sub-clause 10.3.2.3.1.1.1.

Event Control (mandatory): Identifies additional control options for the event. Refer to sub-clause 10.3.2.3.1.1.1.

Signature Type (mandatory): An 8-bit Unsigned integer enumerating the type of algorithm use to create the Signature. The enumerated values are shown in Table 10-28:

Table 10-28. Enumerated Values of Signature Types

Enumerated Value	Signature Type
0x00	No Signature
0x01	ECDSA

If the signature field is not used, the signature type shall be set to 0x00, which will be used to indicate "no signature." The signature field shall be filled with (48) 0xFF values.

Signature (optional): A non-repudiation signature created by using the Matyas-Meyer-Oseas hash function (specified in Annex B.6 in [Z1]) used in conjunction with ECDSA. The signature creation process will occur in two steps:

1. Pass the first ten fields, which includes all fields up to the Signature field, of the Report Event Status command (listed in Figure 10-16) through ECDSA using the device's ECC Private Key, generating the signature (r,s).

   Note: ECDSA internally uses the MMO hash function in place of the internal SHA-1 hash function.

2. Concatenate ECDSA signature components (r,s) and place into the Signature field within the Report Event Status command.

   Note: the lengths of r and s are implicit, based on the curve used. Verifying the signature will require breaking the signature field back into the discrete components r and s, based on the length.

10.3.3.3.1.2 When Generated

This command is generated when the client device detects a change of state for an active Load Control event. (The transmission of this command should be delayed after a random delay between 0 and 5 seconds, to avoid a potential storm of packets.)

10.3.3.3.2 Get Scheduled Events Command

Note: The handling of this command is currently under review, and is likely to change in the next revision of the specification. Refer to CCB 1297 (and associated document 12-0180-00) for further information.

This command is used to request that all scheduled Load Control Events, starting at or after the supplied Start Time, are re-issued to the requesting device. When received by the Server, one or more Load Control Event commands (see sub-clause 10.3.2.3.1) will be sent covering both active and scheduled Load Control Events.

10.3.3.3.2.1 Payload Format

The Get Scheduled Events command payload shall be formatted as illustrated in Figure 10-17.

Figure 10-17. Format of the *Get Scheduled Events* Command Payload

Octets	4	1
Data Type	UTC	uint8
Field Name	Start Time (M)	Number of Events (M)

Start Time (mandatory): UTC Timestamp representing the minimum ending time for any scheduled or currently active events to be resent. If either command has a Start Time of 0x00000000, replace that Start Time with the current time stamp.

Number of Events (mandatory): Represents the maximum number of events to be sent. A value of 0 would indicate all available events are to be returned. Example: Number of Events = 1 would return the first event with an EndTime greater than or equal to the value of Start Time field in the Get Scheduled Events command (EndTime would be StartTime plus Duration of the event listed in the device's event table).

10.3.3.3.2.2 When Generated

This command is generated when the client device wishes to verify the available Load Control Events or after a loss of power/reset occurs and the client device needs to recover currently active or scheduled Load Control Events.

A ZCL Default Response with status NOT_FOUND shall be returned when there are no events available.

10.3.3.4 Commands Received

The client receives the cluster-specific commands detailed in sub-clause 10.3.1.1.

10.3.3.5 Attribute Reporting

Attribute reporting is not expected to be used for this cluster. The Client side attributes are not expected to be changed by the Client, only used during Client operations.

10.3.4 Application Guidelines

The criticality level is sent by the utility to the load control device to indicate how much load reduction is requested. The utility is not required to use all of the criticality levels that are described in this specification. A load control device is not required to provide a unique response to each criticality level that it may receive.

The Average Load Adjustment Percentage, temperature offsets, and temperature set points are used by load control devices and energy management systems on a “voluntary” or “optional” basis. These devices are not required to use the values that are provided by the utility. They are provided as a recommendation by the utility.

The load control device shall, in a manner that is consistent with this specification, accurately report event participation by way of the Report Event Status message.

The Average Load Adjustment Percentage is sent by the utility to the load control device to indicate how much load reduction is requested. The load control device may respond to this information in a unique manner as defined by the device manufacturer.

The Duty Cycle is sent by the utility to the load control device to indicate the maximum “On state” for a device. The control device may respond to this information in a unique manner as defined by the device manufacturer.

The cooling temperature offset may be sent by the utility to the load shed control to indicate how much indoor cooling temperature offset is requested. Response of a load control device to this information is not mandatory. The control device may respond to this information in a unique manner as defined by the device manufacturer.

The heating temperature offset may be sent by the utility to the load control device to indicate how much indoor heating temperature offset is requested. The control device may respond to this information in a unique manner as defined by the device manufacturer.

The cooling temperature may be sent by the utility to the load control device to indicate the indoor cooling temperature setting that is requested. The control device may respond to this information in a unique manner as defined by the device manufacturer.

The heating temperature may be sent by the utility to the load control device to indicate the indoor heating temperature setting that is requested. The control device may respond to this information in a unique manner as defined by the device manufacturer.

Note: The most recent Load Control Event supersedes any previous Load Control Event command for the set of Device Classes and groups for a given time. Nested events and overlapping events are not allowed. The current active event will be terminated if a new event is started.

10.3.4.1 Load Control Rules, Server

10.3.4.1.1 Load Control Server, Identifying Use of SetPoint and Offset Fields

The use of the fields, Heating and Cooling Temperature Set Points and Heating and Cooling Temperature Offsets is optional. All fields in the payload must be populated. Non-use of these fields by the Server is indicated by using the following values: 0x8000 for Set Points and 0xFF for Offsets. When any of these four fields are indicated as optional, they shall be ignored by the client.

10.3.4.1.2 Load Control Server, Editing of Scheduled Events

Editing of a scheduled demand response event is not allowed. Editing of an active demand response event is not allowed. Nested events and overlapping events are not allowed. The current active event will be terminated if a new event is started.

10.3.4.2 Load Control Rules, Client

10.3.4.2.1 Start and Stop Randomization

When shedding loads (turning a load control device off), the load control device will optionally apply start time randomization based on the values specified in the Event Control Bits and the Client's Start Randomization Minutes attribute. By default, devices will apply a random delay as specified by the default values of start and end randomization in the Demand Response Client Cluster Attributes table.

When ending a load control event, the load control device will support the same randomization features as provided in the start load control event.

10.3.4.2.2 Editing of DR Control Parameters

In Load Control Device and energy management systems, editing of the demand response control parameters while participating in an active demand response event is not allowed.

10.3.4.2.3 Response to Price Events + Load Control Events

The residential system's response to price driven events will be considered in addition to the residential system's response to demand response events. Demand response events which require that the residential system is turned off have priority over price driven events. Demand response events which require that the residential system go to a fixed setting point have priority over price driven events. In this case, the thermostat shall not use a Cooling or Heating Temperature Set Point that causes the device to use more energy than the price driven event setting.

10.3.4.2.4 Opt-Out Messages

An event override message, "opt-out", will be sent by the load control device or energy management system if the operator chooses not to participate in a demand response event by taking action to override the programmed demand reduction response. The override message will be sent at the start of the event. In the case where the event has been acknowledged and started, the override message will be sent when the override occurs.

10.3.4.2.5 Thermostat/HVAC Controls

A residential HVAC system will be allowed to change mode, from off to Heat, off to Cool, Cool to Heat, or Heat to Cool, during a voluntary event which is currently active. The HVAC control must acknowledge the event, as if it was operating, in that mode, at the start of the event. The HVAC control must obey the event rules that would have been enforced if the system had been operating in that mode at the start of the active event.

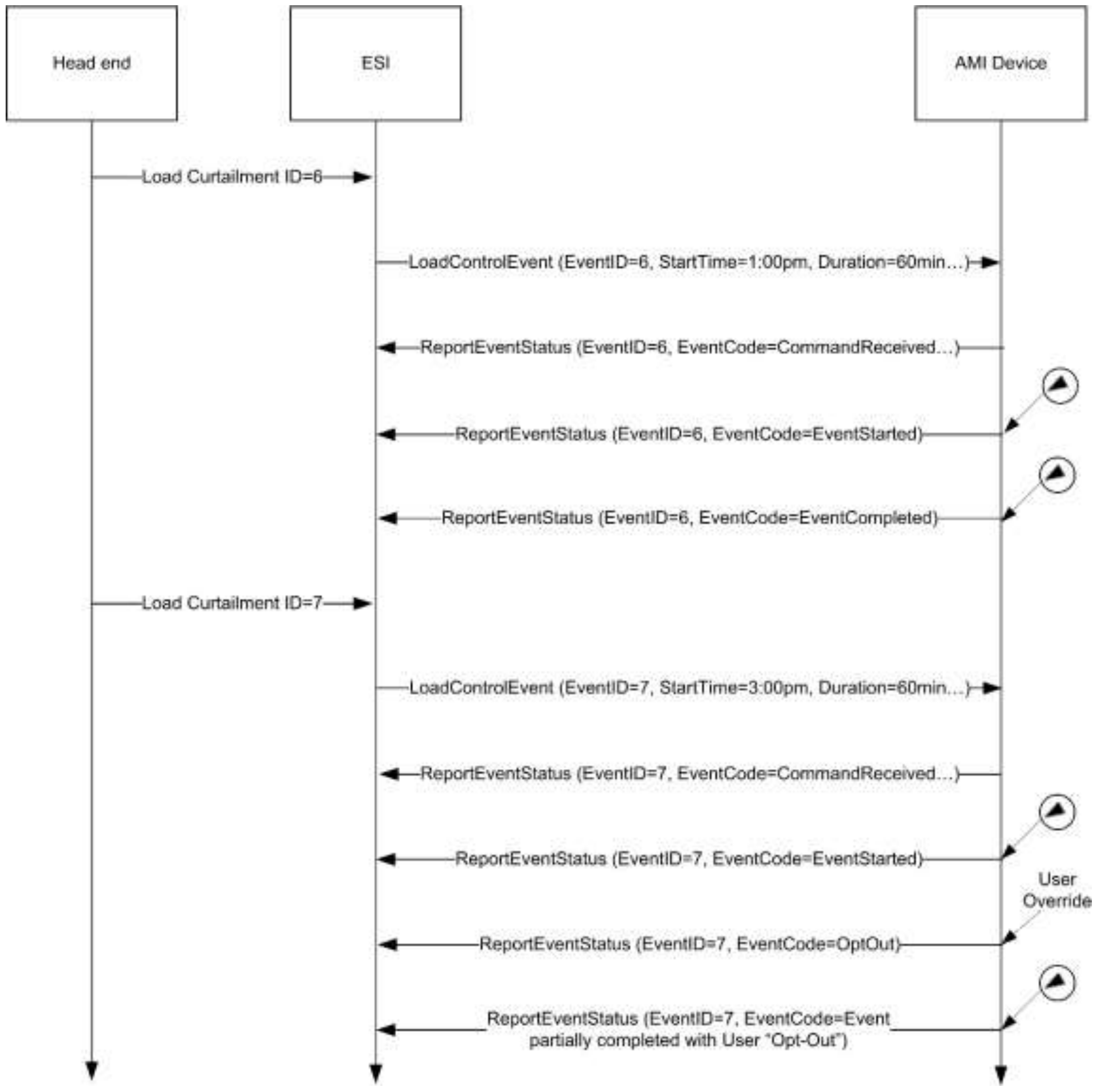
An event override message, “opt-out”, will be sent by the load control device or energy management system if the operator chooses not to participate in a demand response event by taking action to override the programmed demand reduction response. The override message will be sent at the start of the event. In the case where the event has been acknowledged and started, the override message will be sent when the override occurs.

10.3.4.2.6 Demand Response and Load Control Transaction Examples

The example in Figure 10-18 depicts the transactions that would take place for two events, one that is successful and another that is overridden by the user.

13691 **Figure 10-18. Example of Both a Successful and an Overridden Load Curtailment Event**

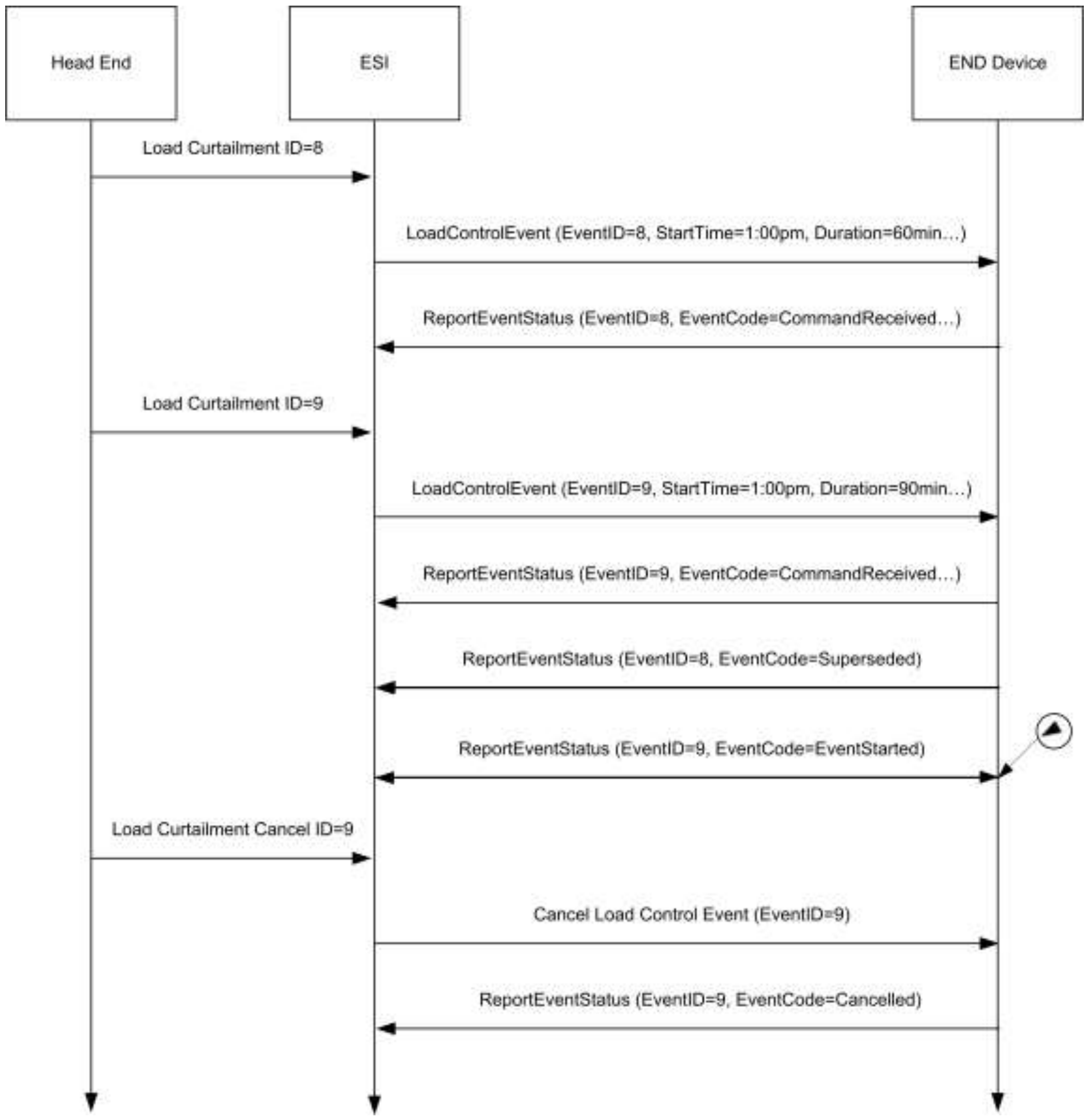
13692


Head end
ESI
AMI Device
Load Curtailment ID=6
LoadControlEvent (EventID=6, StartTime=1:00pm, Duration=60min…)
ReportEventStatus (EventID=6, EventCode=CommandReceived…)
ReportEventStatus (EventID=6, EventCode=EventStarted)
ReportEventStatus (EventID=6, EventCode=EventCompleted)
Load Curtailment ID=7
LoadControlEvent (EventID=7, StartTime=3:00pm, Duration=60min…)
ReportEventStatus (EventID=7, EventCode=CommandReceived…)
ReportEventStatus (EventID=7, EventCode=EventStarted)
User Override
ReportEventStatus (EventID=7, EventCode=OptOut)
ReportEventStatus (EventID=7, EventCode=Event partially completed with User "Opt-Out")

13693

13694 The example in Figure 10-19 depicts the transactions that would take place when an event is superseded by an event
13695 that is eventually cancelled.

Figure 10-19. Example of a Load Curtailment Superseded and Another Cancelled


Head End
ESI
END Device
Load Curtailment ID=8
LoadControlEvent (EventID=8, StartTime=1:00pm, Duration=60min…)
ReportEventStatus (EventID=8, EventCode=CommandReceived…)
Load Curtailment ID=9
LoadControlEvent (EventID=9, StartTime=1:00pm, Duration=90min…)
ReportEventStatus (EventID=9, EventCode=CommandReceived…)
ReportEventStatus (EventID=8, EventCode=Superseded)
ReportEventStatus (EventID=9, EventCode=EventStarted)
Load Curtailment Cancel ID=9
Cancel Load Control Event (EventID=9)
ReportEventStatus (EventID=9, EventCode=Cancelled)

Refer to section 10.3.5 for more information regarding the management and behavior of overlapping events.

10.3.5 Rules and Guidelines for Overlapping Events

This section describes multiple scenarios that Demand Response and Load Control devices may encounter over the Smart Energy network. The examples describe situations of overlapping events that are acceptable and where overlapping events that will be superseded due to conflicts.

10.3.5.1 Definitions

Start Time – "Start Time" field contained within the Load Control Event packet indicating when the event should start. Please note, a "Start Time" value of 0x00000000 denotes "now" and the device should use its current time as the "Start Time."

Duration – "Duration" field contained within the Load Control Event packet indicating how long the event should occur.

End Time – Time when Event completes as calculated by adding Duration to Start Time.

Scheduled Period – Represents the time between the Start Time and the End Time of the event.

Effective Start Time -Represents time at which a specific device starts a load control event based on the Start Time plus or minus any randomization offsets.

Effective End Time – Represents time at which a specific device ends a load control event based on the Start Time plus Duration, plus or minus any randomization offsets.

Effective Scheduled Period – Represents the time between the Effective Start Time and the Effective End Time.

Overlapping Event – Defined as an event where the Scheduled Period covers part or all of an existing, previously scheduled event.

Successive Events – Defined as two events where the scheduled End Time of the first event is equal the Start Time of a subsequent scheduled event.

Nested Events – Defined as two events where the scheduled Start Time and End Time of the second event falls during the Scheduled Period of the first scheduled event and the second event is of shorter duration than the first event.

10.3.5.2 Rules and Guidelines

The depicted behaviors and required application management decisions are driven from the following guidance and rule set:

1. Upstream Demand Response/Load Control systems and/or the ESI shall prevent mismanaged scheduling of *Overlapping Events* or *Nested Events*. It is recognized Upstream Demand Response/Load Control systems and/or the ESI will need to react to changing conditions on the grid by sending *Overlapping Events* or *Nested Events* to supersede previous directives. But those systems must have the proper auditing and management rules to prevent a cascading set of error conditions propagated by improperly scheduled events.
2. When needed, Upstream Demand Response/Load Control systems and/or the ESI may resolve any event scheduling conflicts by performing one of the following processes:
    a. Canceling individual events starting with the earliest scheduled event and re-issuing a new set of events.
    b. Canceling all scheduled events and re-issuing a new set of events.
    c. Sending *Overlapping Events* or *Nested Events* to supersede previous directives.

    It is recommended that process 2.c is used for most situations since it can allow a smoother change between two sets of directives, but no way does it negate the responsibilities identified in rule #1.
3. When an End Device receives an event with the *End Time* in the past (*End Time* < Current Time), this event is ignored and a *Report Event Status* command is returned with the Event Status set to 0xFB (Rejected - Event was received after it had expired).
4. When an End Device receives an event with a Start Time in the past and an End Time in the future ((Start Time < Current Time) AND (End Time > Current Time)), the event is processed immediately. The Effective Start Time is calculated using the Current Time as the Start Time. Original End Time is preserved.
5. Regardless of the state of an event (scheduled or executing), when an End Device detects an Overlapping Event condition the latest Overlapping Event will take precedence over the previous event. Depending on the state of the event (scheduled or executing), one of the following steps shall take place:
    a. If the previous event is scheduled and not executing, the End Device returns a Report Event Status command (referencing the previous event) with the Event Status set to 0x07 (The event has been

superseded). After the *Report Event Status* command is successfully sent, the End Device can remove the previous event schedule.

   b. If the previous event is executing, the End Device shall change directly from its current state to the requested state at the *Effective Start Time* of the *Overlapping Event* (Note: Rule #4 effects *Effective Start Time*). The End Device returns a *Report Event Status* command (referencing the previous event) with the Event Status set to 0x07 (the event has been superseded).

6. Randomization ***shall not*** cause event conflicts or unmanaged gaps. To clarify:

   a. When event starting randomization is requested, time periods between the *Start Time* of an event and the *Effective Start Time* a device should either maintain its current state or apply changes which contribute to energy saving. Preference would be to maintain current state.

   b. When event ending randomization is used and the *Effective End Time* overlaps the *Effective Start Time* of a *Successive Event*, the *Effective Start Time* takes precedence. Events are not reported as superseded, End devices should report event status as it would a normal set of *Successive Events.*

   c. It is recommended devices apply the same Start and Stop Randomization values for consecutive events to help prevent unexpected gaps between events.

   d. Devices ***shall not*** artificially create a gap between *Successive Events.*

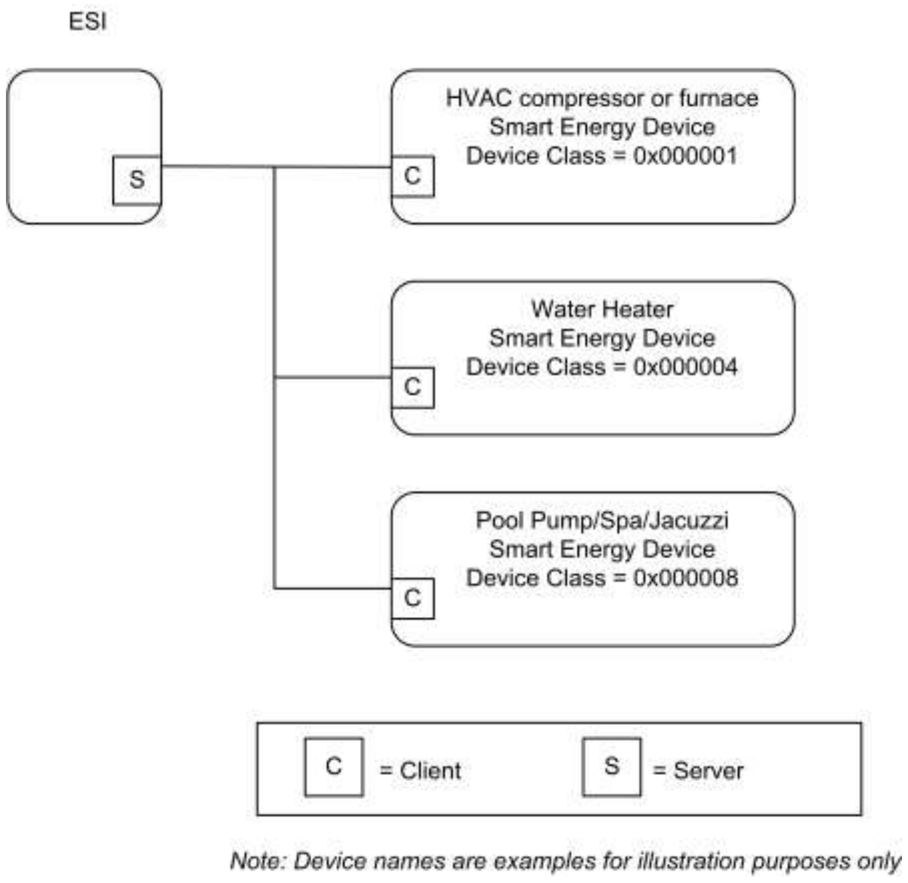
7. It is permissible to have gaps when events are not Successive Events or Overlapping Events.

8. If multiple device classes are identified for an event, future events for individual device classes (or a subset of the original event) that cause an Overlapping Event will supersede the original event strictly for that device class (or a subset of the original event). Note: Rule #5 applies to all Overlapping Events.

10.3.5.3 Event Examples

Smart Energy devices which act upon Demand Response and Load Control events shall use the following examples for understanding and managing overlapping and superseded events. Within those examples, references to multiple device classes will be used. Figure 10-20 depicts a representation of those devices in a Smart Energy network.

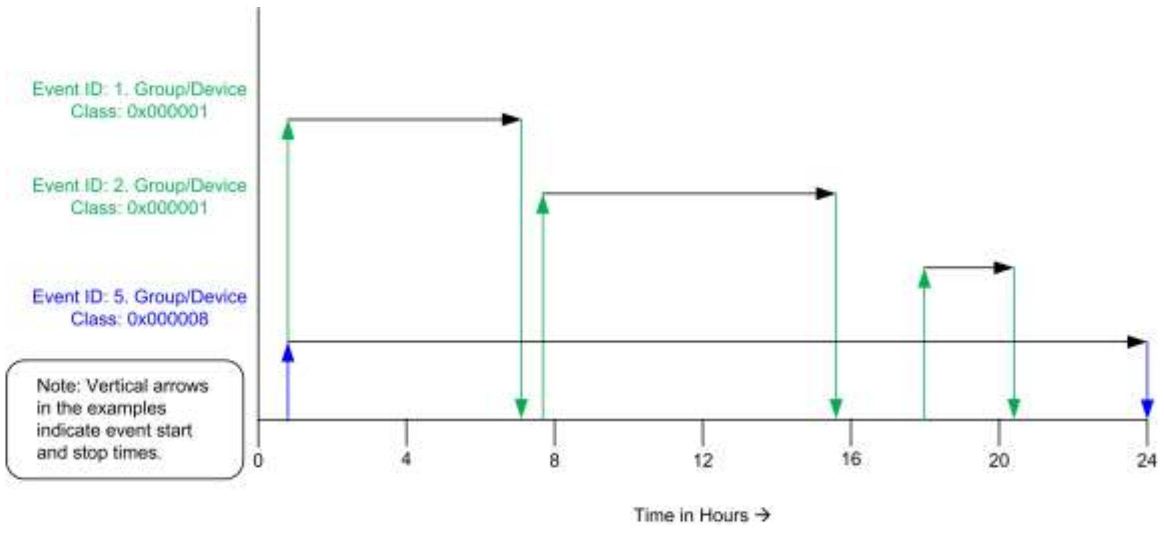
Figure 10-20. Smart Energy Device Class Reference Example



10.3.5.3.1 Correct Overlapping Events for Different Device Classes

Figure 10-21 depicts a correct series of DR/LC event for device class of 0x000001 (reference for the BitMap definition) with an event scheduled for another device class during the same period.

Figure 10-21. Correctly Overlapping Events

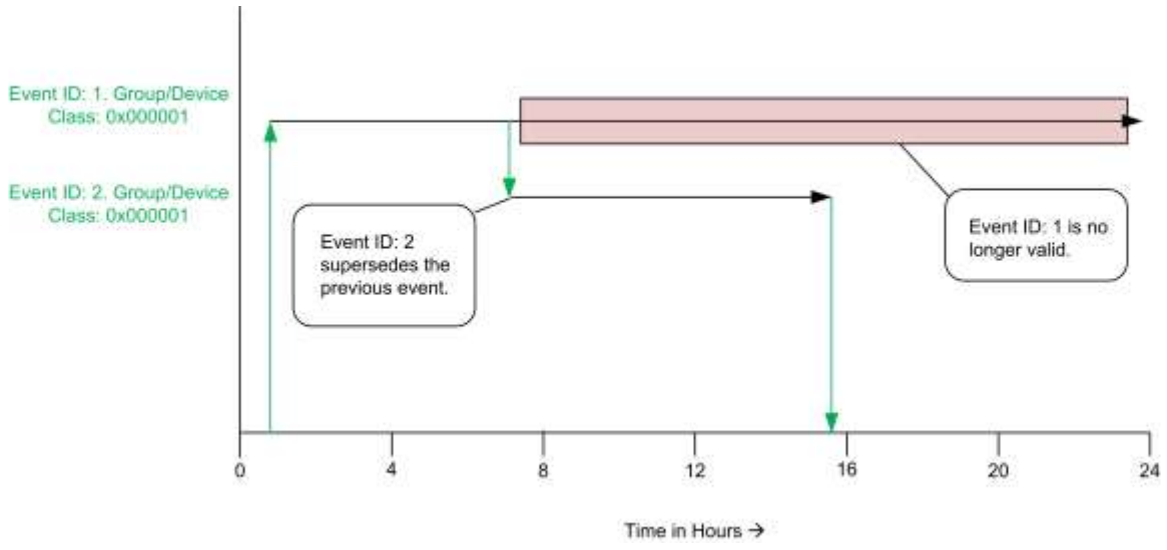


In Figure 10-21, Device Class 0x000001 receives a sequence of 3 unique DR/LC events to be scheduled and acted upon. During this same 24 hour period, Device Class 0x000008 receives one scheduled DR/LC event that spans across the same time period as the events scheduled for Device Class 0x0000001. Because both Device Classes are unique, there are no conflicts due to Overlapping Events.

10.3.5.3.2 Correct Superseded Event for a Device Class

Figure 10-22 depicts a correct series of DR/LC events for device class of 0x000001 (reference for the BitMap definition) where an event is scheduled then later superseded.

Figure 10-22. Correct Superseding of Events

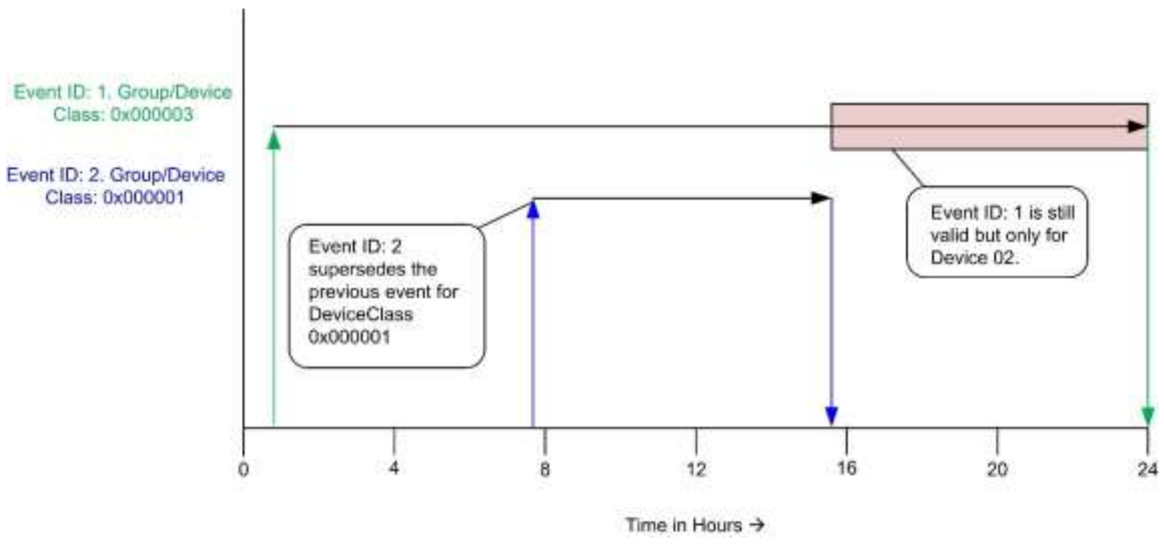

Event ID: 1. Group/Device Class: 0x000001
Event ID: 2. Group/Device Class: 0x000001
Event ID: 2 supersedes the previous event.
Event ID: 1 is no longer valid.
0
4
8
12
16
20
24
Time in Hours →

In Figure 10-22, Device Class 0x000001 receives DR/LC Event ID#1 setup for a 24 hour Scheduled Period, which later is superseded by DR/LC Event ID#2, invalidating the remainder of Event ID#1, which is cancelled.

10.3.5.3.3 Superseding Events for Subsets of Device Classes

Figure 10-23 depicts a correct series of DR/LC events for device class of 0x000001 (reference for the BitMap definition) with an event scheduled for another device class during the same time period.

Figure 10-23. Superseded Event for a Subset of Device Classes

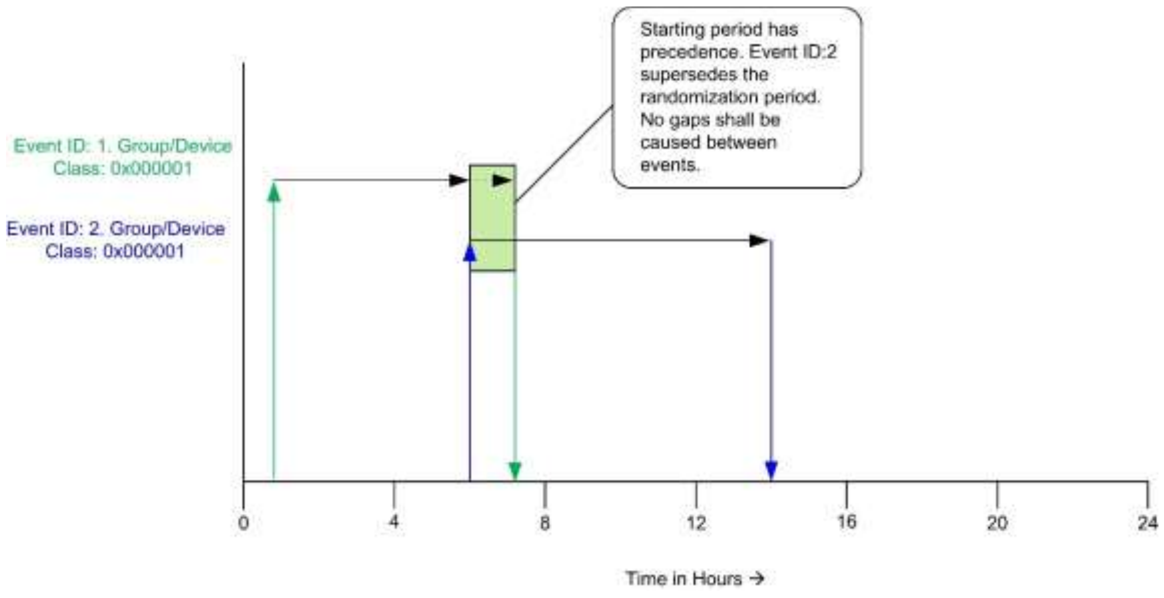

Event ID: 1. Group/Device Class: 0x000003
Event ID: 2. Group/Device Class: 0x000001
Event ID: 2 supersedes the previous event for DeviceClass 0x000001
Event ID: 1 is still valid but only for Device 02.
0
4
8
12
16
20
24
Time in Hours →

In Figure 10-23, Device Class 0x000003 receives DR/LC Event ID#1 setup for a 24 hour Scheduled Period, which is targeted for both Device Class 0x000002 and 0x000001 (OR'ed == 0x000003). In the example, Event ID#2 is issued only for Device Class 0x000001, invalidating the remainder of Event ID#1 for that device class. DR/LC Event ID#1 is still valid for Device Class 0x000002, which in the example should run to completion.

10.3.5.3.4 Ending Randomization Between Events

Figure 10-24 depicts an Effective End Time that overlaps a second scheduled DR/LC event for device class of 0x000001 (reference for the BitMap definition).

Figure 10-24. Ending Randomization Between Events

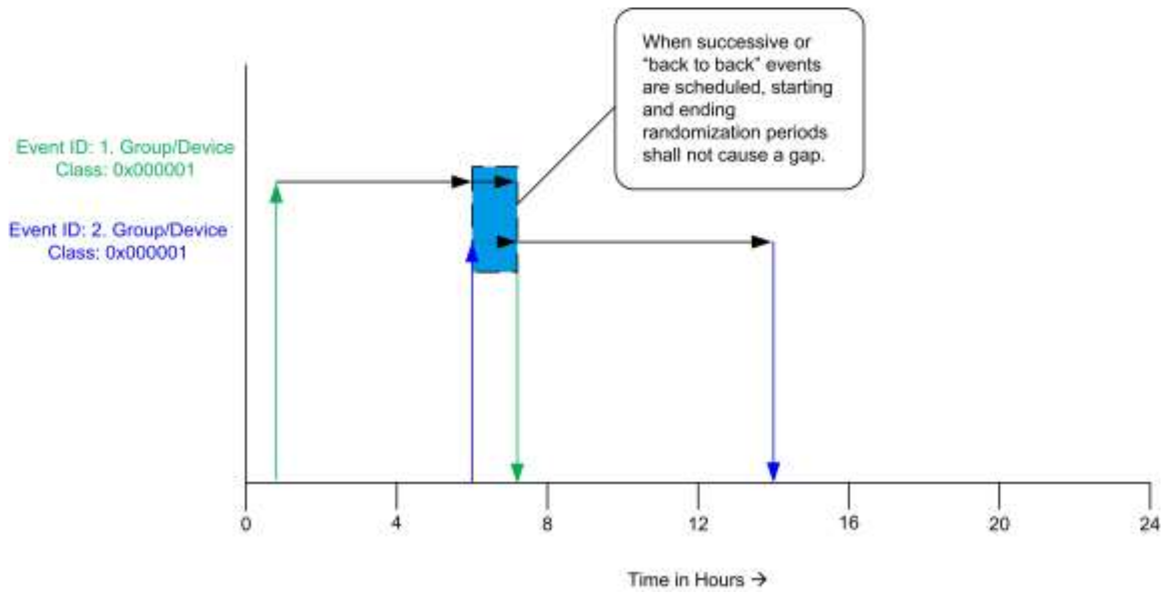

Event ID: 1. Group/Device Class: 0x000001
Event ID: 2. Group/Device Class: 0x000001
Starting period has precedence. Event ID:2 supersedes the randomization period. No gaps shall be caused between events.
0
4
8
12
16
20
24
Time in Hours →

In Figure 10-24, Device Class 0x000001 receives a DR/LC Event ID#1 with an ending randomization setting (please refer to sub-clause 10.3.2.3.1.1.1 for more detail). A second DR/LC (Event ID#2) is issued with a starting time which matches the ending time of DR/LC Event ID#1. In this situation, the Start Time of Event ID#2 has precedence. Event ID#1 is not reported as superseded.

10.3.5.3.5 Start Randomization Between Events

Figure 10-25 depicts an Effective Start Time that overlaps a previously scheduled DR/LC event for device class of 0x000001 (reference for the BitMap definition).

Figure 10-25. Start Randomization Between Events

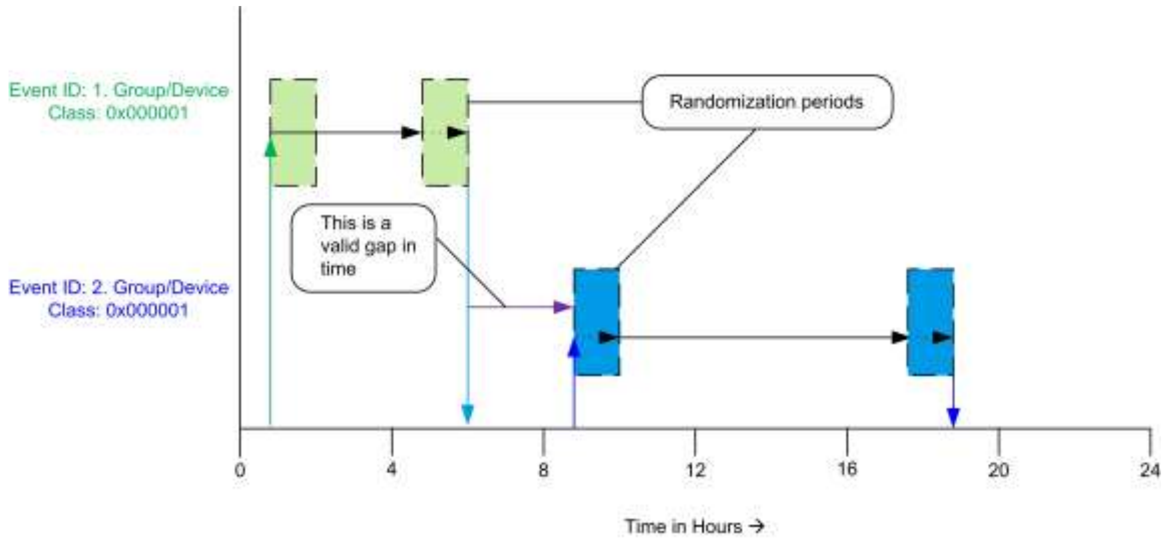

When successive or "back to back" events are scheduled, starting and ending randomization periods shall not cause a gap.
Event ID: 1. Group/Device Class: 0x000001
Event ID: 2. Group/Device Class: 0x000001
0
4
8
12
16
20
24
Time in Hours →

In Figure 10-25, Device Class 0x000001 receives a DR/LC Event ID#1 with an ending randomization setting (please refer to sub-clause 10.3.2.3.1.1.1 for more detail). Effective End Time of Event ID#1 is not known. A second DR/LC (Event ID#2) is issued with a starting randomized setting, which has an Effective Start Time that could overlap or start after the Effective End Time of DR/LC Event ID#1. In this situation, the Effective Start Time of Event ID#2 has precedence but the DR/LC device must also prevent any artificial gaps caused by the Effective Start Time of Event ID#2 and Effective End Time of Event ID#1.

10.3.5.3.6 Acceptable Gaps Caused by Start and Stop Randomization of Events

Figure 10-26 depicts an acceptable gap between two scheduled DR/LC events for device class of 0x000001 (reference for the BitMap definition) using both starting and ending randomization with both events.

Figure 10-26. Acceptable Gaps with Start and Stop Randomization



In Figure 10-26, Device Class 0x000001 receives a DR/LC Event ID#1 with both a starting and ending randomization setting. (Please refer to sub-clause 10.3.2.3.1.1.1 for more detail). A second DR/LC Event ID#2 is also issued with both a starting and ending randomized setting. The primary configuration to note in this example is the Effective End Time of DR/LC Event ID#1 completes well in advance of the Effective Start Time of DR/LC Event ID#2. In this scenario, regardless of randomization a gap is naturally created by the scheduling of the events and is acceptable.

10.4 Metering

10.4.1 Overview

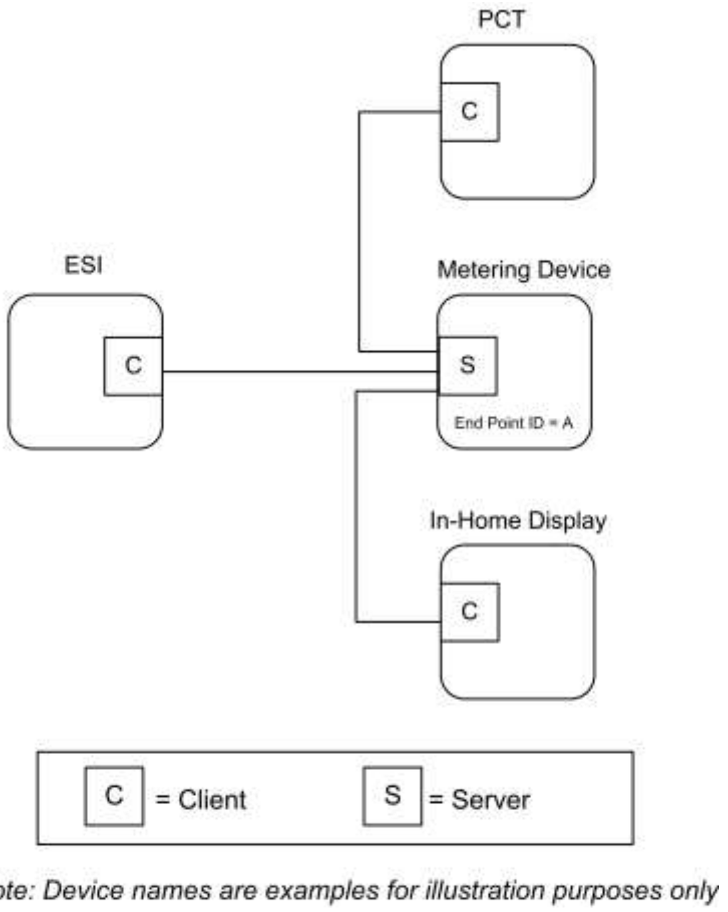
Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The Metering Cluster provides a mechanism to retrieve usage information from Electric, Gas, Water, and potentially Thermal metering devices. These devices can operate on either battery or mains power, and can have a wide variety of sophistication. The Metering Cluster is designed to provide flexibility while limiting capabilities to a set number of metered information types. More advanced forms or data sets from metering devices will be supported in the Smart Energy Tunneling Cluster, which will be defined in sub-clause 10.6.

The following figures identify three configurations as examples utilizing the Metering Cluster.

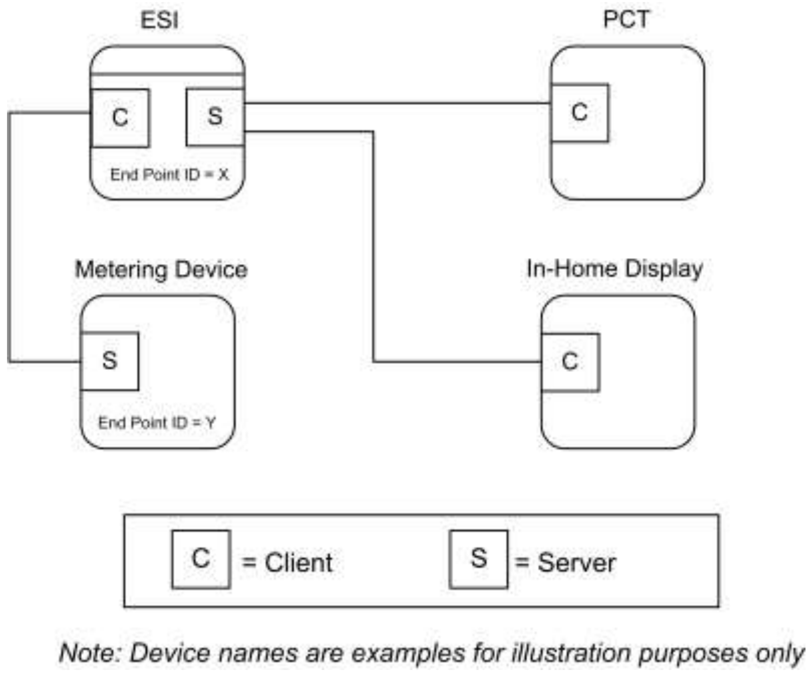
In Figure 10-27, the metering device is the source of information provided via the Metering Cluster Server.

Figure 10-27. Standalone ESI Model with Mains Powered Metering Device



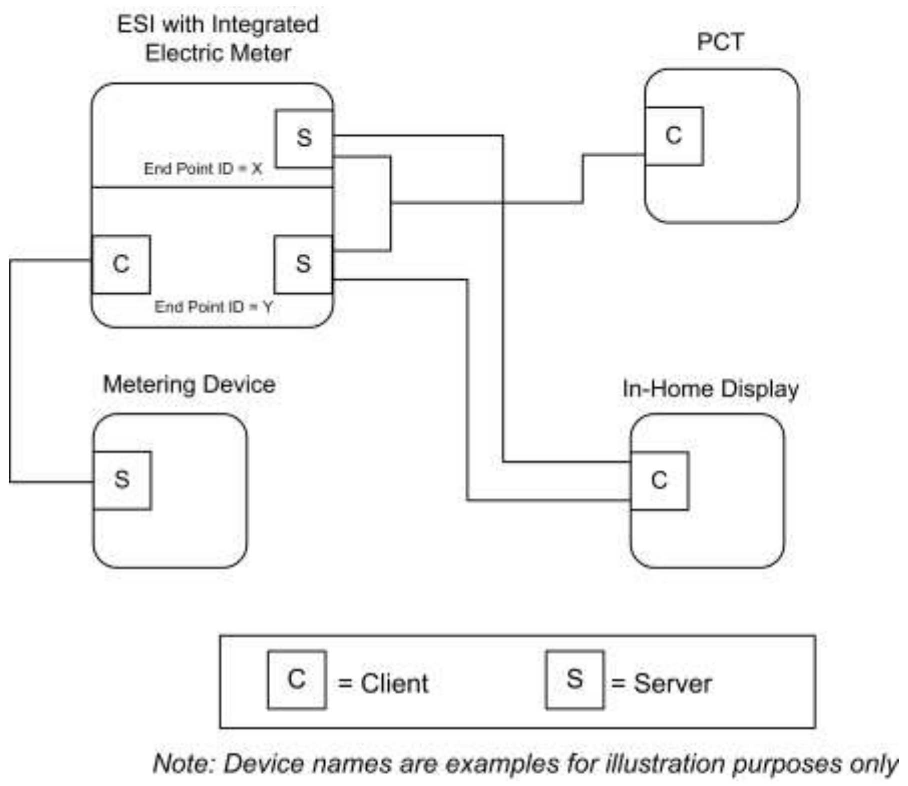
In the example shown in Figure 10-28, the metering device is running on battery power and its duty cycle for providing information is unknown. It’s expected the ESI will act like a mirrored image or a mailbox (Client) for the metering device data, allowing other Smart Energy devices to gain access to the metering device’s data (provided via an image of its Metering Cluster).

Figure 10-28. Standalone ESI Model with Battery Powered Metering Device



In the example shown in Figure 10-29, much like the previous example in Figure 10-28, the external metering device is running on battery power and its duty cycle for providing information is unknown. It’s expected the ESI will act like a Client side mailbox for the external metering device data, allowing other Smart Energy devices to gain access to the metering device’s data (provided via an image of its Metering Cluster). Since the ESI can also contain an integrated metering device where its information is also conveyed through the Metering Cluster, each device (external metering device mailbox and integrated meter) will be available via independent EndPoint IDs. Other Smart Energy devices that need to access the information must understand the ESI cluster support by performing service discoveries. It can also identify if an Endpoint ID is a mailbox/ mirror of a metering device by reading the MeteringDeviceType attribute (refer to sub-clause 10.4.2.2.4.7).

Figure 10-29. ESI Model with Integrated Metering Device


ESI with Integrated Electric Meter
S
End Point ID = X
C
S
End Point ID = Y
PCT
C
Metering Device
S
In-Home Display
C
C = Client
S = Server
Note: Device names are examples for illustration purposes only

In the above examples (Figure 10-28 and Figure 10-29), it's expected the ESI would perform Attribute Reads (or configure Attribute Reporting) and use the GetProfile command to receive the latest information whenever the Metering Device (EndPoint Z) wakes up. When received, the ESI will update its mailbox (EndPoint ID Y in Figure 10-28 and Figure 10-29) to reflect the latest data available. A metering device using the mirror is also allowed (and recommended) to push metering data updates to the ESI via Report Attribute commands as described in sub-clause 10.4.3.4.

Other Smart Energy devices can access EndPoint Y in the ESI to receive the latest information just as they would to access information in the ESI's integrated Electric meter (as in Figure 10-29, EndPoint X) and other Metering devices (as in Figure 10-27, EndPoint A).

10.4.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

10.4.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	SEMT	Type 1 (client to server)

10.4.1.3 Cluster Identifiers

Identifier	Name
0x0702	Metering (Smart Energy)

10.4.2 Server

10.4.2.1 Dependencies

Subscribed reporting of Metering attributes.

10.4.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 256 attributes. Attribute identifiers are encoded such that the most significant octet specifies the attribute set and the least significant octet specifies the attribute within the set. The currently defined attribute sets are listed in Table 10-29.

Note: Certain attributes within this cluster are provisionary and not certifiable. Refer to the individual attribute sets for details of the relevant attributes.

Table 10-29. Metering Cluster Attribute Sets

Attribute Set Identifier	Description
0x00	Reading Information Set
0x01	TOU Information Set
0x02	Meter Status
0x03	Formatting
0x04	Historical Consumption
0x05	Load Profile Configuration
0x06	Supply Limit
0x07	Block Information
0x08	Alarms

10.4.2.2.1 Reading Information Set

The set of attributes shown in Table 10-30 provides a remote access to the reading of the Electric, Gas, or Water metering device. A reading must support at least one register which is the actual total summation of the delivered quantity (kWh, m^3, ft^3, ccf, US gl).

Please note: In the following attributes, the term “Delivered” refers to the quantity of Energy, Gas, or Water that was delivered to the customer from the utility. Likewise, the term “Received” refers to the quantity of Energy, Gas, or Water that was received by the utility from the customer.

Note: Metering Cluster Reading Attributes 0x10-0x14 in this revision of this specification are provisionary and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

13902 **Table 10-30. Reading Information Attribute Set**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*CurrentSummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	M
0x0001	*CurrentSummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0002	*CurrentMaxDemandDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0003	*CurrentMaxDemandReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0004	*DFTSummation*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0005	*DailyFreezeTime*	uint16	0x0000 to 0x183C	R	0x0000	O
0x0006	*PowerFactor*	int8	-100 to +100	R	0x00	O
0x0007	*ReadingSnapShotTime*	UTC		R	-	O
0x0008	*CurrentMaxDemandDeliveredTime*	UTC		R	-	O
0x0009	*CurrentMaxDemandReceivedTime*	UTC		R	-	O
0x000A	*DefaultUpdatePeriod*	uint8	0x00 to 0xFF	R	0x1E	O
0x000B	*FastPollUpdatePeriod*	uint8	0x00 to 0xFF	R	0x05	O
0x000C	*CurrentBlockPeriodConsumptionDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x000D	*DailyConsumptionTarget*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x000E	*CurrentBlock*	enum8	0x00 to 0x10	R	-	O
0x000F	*ProfileIntervalPeriod*	enum8	0x00 to 0xFF	R	-	O
0x0010	*IntervalReadReportingPeriod*	uint16	0x0000 to 0xFFFF	R	0	O
0x0011	*PresetReadingTime*	uint16	0x0000 to 0x173B	R	0x0000	O
0x0012	*VolumePerReport*	uint16	0x0000 to 0xFFFF	R	-	O
0x0013	*FlowRestriction*	uint8	0x00 to 0xFF	R	-	O

Id	Name	Type	Range	Acc	Def	M/O
0x0014	*Supply Status*	enum8	0x00 to 0xFF	R	-	O
0x0015	*CurrentInletEnergyCarrierSummation*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	M*
0x0016	*CurrentOutletEnergyCarrierSummation*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0017	*InletTemperature*	int24	-8,388,607 to	R	-	M*
0x0018	*OutletTemperature*	int24	-8,388,607 to 8,388,607	R	-	M*
0x0019	*ControlTemperature*	int24	-8,388,607 to 8,388,607	R	-	O
0x001A	*CurrentInletEnergyCarrierDemand*	int24	-8,388,607 to 8,388,607	R	-	O
0x001B	*CurrentOutletEnergyCarrierDemand*	int24	-8,388,607 to 8,388,607	R	-	O
0x001C	*PreviousBlockPeriodConsumptionDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O

** Mandatory for Heat or Cooling; Optional for others*

10.4.2.2.1.1 *CurrentSummationDelivered* Attribute

CurrentSummationDelivered represents the most recent summed value of Energy, Gas, or Water delivered and consumed in the premises. CurrentSummationDelivered is mandatory and must be provided as part of the minimum data set to be provided by the metering device. CurrentSummationDelivered is updated continuously as new measurements are made.

10.4.2.2.1.2 *CurrentSummationReceived* Attribute

CurrentSummationReceived represents the most recent summed value of Energy, Gas, or Water generated and delivered from the premises. If optionally provided, CurrentSummationReceived is updated continuously as new measurements are made.

10.4.2.2.1.3 *CurrentMaxDemandDelivered* Attribute

CurrentMaxDemandDelivered represents the maximum demand or rate of delivered value of Energy, Gas, or Water being utilized at the premises. If optionally provided, CurrentMaxDemandDelivered is updated continuously as new measurements are made.

10.4.2.2.1.4 *CurrentMaxDemandReceived* Attribute

CurrentMaxDemandReceived represents the maximum demand or rate of received value of Energy, Gas, or Water being utilized by the utility. If optionally provided, CurrentMaxDemandReceived is updated continuously as new measurements are made.

10.4.2.2.1.5 *DFTSummation* Attribute

DFTSummation represents a snapshot of attribute CurrentSummationDelivered captured at the time indicated by attribute DailyFreezeTime. If optionally provided, DFTSummation is updated once every 24 hours and captured at the time set in sub-clause 10.4.2.2.1.6.

10.4.2.2.1.6 *DailyFreezeTime* Attribute

DailyFreezeTime represents the time of day when DFTSummation is captured. DailyFreezeTime is an unsigned 16-bit value representing the hour and minutes for DFT. The byte usages are:

Bits 0 to 7: Range of 0 to 0x3C representing the number of minutes past the top of the hour.

Bits 8 to 15: Range of 0 to 0x17 representing the hour of the day (in 24-hour format).

10.4.2.2.1.7 *PowerFactor* Attribute

PowerFactor contains the Average Power Factor ratio in 1/100ths. Valid values are 0 to 99.

10.4.2.2.1.8 *ReadingSnapShotTime* Attribute

The ReadingSnapShotTime attribute represents the last time all of the CurrentSummationDelivered, CurrentSummationReceived, CurrentMaxDemandDelivered, and CurrentMaxDemandReceived attributes that are supported by the device were updated.

10.4.2.2.1.9 *CurrentMaxDemandDeliveredTime* Attribute

The CurrentMaxDemandDeliveredTime attribute represents the time when CurrentMaxDemandDelivered reading was captured.

10.4.2.2.1.10 *CurrentMaxDemandReceivedTime* Attribute

The CurrentMaxDemandReceivedTime attribute represents the time when CurrentMaxDemandReceived reading was captured.

10.4.2.2.1.11 *DefaultUpdatePeriod* Attribute

The DefaultUpdatePeriod attribute represents the interval (seconds) at which the InstantaneousDemand attribute is updated when not in fast poll mode. InstantaneousDemand may be continuously updated as new measurements are acquired, but at a minimum InstantaneousDemand must be updated at the DefaultUpdatePeriod. The DefaultUpdatePeriod may apply to other attributes as defined by the device manufacturer.

10.4.2.2.1.12 *FastPollUpdatePeriod* Attribute

The FastPollUpdatePeriod attribute represents the interval (seconds) at which the InstantaneousDemand attribute is updated when in fast poll mode. InstantaneousDemand may be continuously updated as new measurements are acquired, but at a minimum, InstantaneousDemand must be updated at the FastPollUpdatePeriod. The FastPollUpdatePeriod may apply to other attributes as defined by the device manufacturer.

10.4.2.2.1.13 *CurrentBlockPeriodConsumptionDelivered* Attribute

The CurrentBlockPeriodConsumptionDelivered attribute represents the most recent summed value of Energy, Gas or Water delivered and consumed in the premises during the Block Tariff Period.

The CurrentBlockPeriodConsumptionDelivered is reset at the start of each Block Tariff Period.

10.4.2.2.1.14 *DailyConsumptionTarget* Attribute

The DailyConsumptionTarget attribute is a daily target consumption amount that can be displayed to the consumer on a HAN device, with the intent that it can be used to compare to actual daily consumption (e.g. compare to the CurrentDayConsumptionDelivered).

This may be sent from the utility to the ESI, or it may be derived. Although intended to be based on Block Thresholds, it can be used for other targets not related to blocks. The formatting will be based on the HistoricalConsumptionFormatting attribute.

Example: If based on a Block Threshold, the DailyConsumptionTarget could be calculated based on the number of days specified in the Block Tariff Period and a given Block Threshold as follows: DailyConsumptionTarget = BlockNThreshold / ((BlockPeriodDuration /60) / 24). Example: If the target is based on a Block1Threshold of 675kWh and where 43200 BlockThresholdPeriod is the number of minutes in the billing period (30 days), the ConsumptionDailyTarget would be 675 / ((43200 / 60) / 24) = 22.5 kWh per day.

10.4.2.2.1.15 *CurrentBlock* Attribute

When Block Tariffs are enabled, CurrentBlock is an 8-bit Enumeration which indicates the currently active block. If blocks are active then the current active block is based on the CurrentBlockPeriodConsumptionDelivered and the block thresholds. Block 1 is active when the value of CurrentBlockPeriodConsumptionDelivered is less than Block1Threshold value; Block 2 is active when CurrentBlockPeriodConsumptionDelivered is greater than Block1Threshold value and less than Block2Threshold value, and so on. Block 16 is active when the value of CurrentBlockPeriodConsumptionDelivered is greater than Block15Threshold value.

Table 10-31. Block Enumerations

Enumerated Value	Register Block
0x00	No Blocks in use
0x01	Block1
0x02	Block2
0x03	Block3
0x04	Block4
0x05	Block5
0x06	Block6
0x07	Block7
0x08	Block8
0x09	Block9
0x0A	Block10
0x0B	Block11
0x0C	Block12
0x0D	Block13
0x0E	Block14
0x0F	Block15
0x10	Block16

10.4.2.2.1.16 *ProfileIntervalPeriod* Attribute

The ProfileIntervalPeriod attribute is currently included in the Get Profile Response command payload, but does not appear in an attribute set. This represents the duration of each interval. ProfileIntervalPeriod represents the interval or time frame used to capture metered Energy, Gas, and Water consumption for profiling purposes. The enumeration for this field shall match one of the ProfileIntervalPeriod values defined in sub-clause 10.4.2.3.1.1.1.

10.4.2.2.1.17 *IntervalReadReportingPeriod* Attribute

The IntervalReadReportingPeriod attribute represents how often (in minutes) the water or gas meter is to wake up and provide interval data. E.g.: If IntervalReadReportingPeriod is set to 360, then every 6 hours the water or gas meter is to wake up and provide 6 hours of interval data in a Get Profile Response command. If it is set to 5760, then every 4 days it will wake up and provide 4 days of interval data in a Get Profile Response command. In some cases data may overlap data sent in previous Get Profile Response command.

10.4.2.2.1.18 *PresetReadingTime* Attribute

The PresetReadingTime attribute represents the time of day (in quarter hour increments) at which the meter will wake up and report a register reading even if there has been no consumption for the previous 24 hours. PresetReadingTime is an unsigned 16-bit value representing the hour and minutes. The byte usages are:

Bits 0 to 7: Range of 0 to 0x3B representing the number of minutes past the top of the hour.

Bits 8 to 15: Range of 0 to 0x17 representing the hour of the day (in 24-hour format).

E.g.: A setting of 0x172D would represent 23:45 hours or 11:45 pm; a setting of 0x071E would represent 07:30 hours or 7:30 am. A setting of 0xFFFF indicates this feature is disabled. The use of Attribute Reporting Configuration is optional.

10.4.2.2.1.19 *VolumePerReport* Attribute

The VolumePerReport attribute represents the volume per report increment from the water or gas meter. For example a gas meter might be set to report its register reading for every time 1 cubic meter of gas is used. For a water meter it might report the register value every 10 liters of water usage.

10.4.2.2.1.20 *FlowRestriction* Attribute

The FlowRestriction attribute represents the volume per minute limit set in the flow restrictor. This applies to water but not for gas. A setting of 0xFF indicates this feature is disabled.

10.4.2.2.1.21 *SupplyStatus* Attribute

The SupplyStatus attribute represents the state of the supply at the customer's premises. The enumerated values for this field are outlined in Table 10-32.

Table 10-32. Supply Status Attribute Enumerations

Enumerated Value	Status
0x00	Supply OFF
0x01	Supply OFF/ARMED
0x02	Supply ON

10.4.2.2.1.22 *CurrentInletEnergyCarrierSummation* Attribute

CurrentInletEnergyCarrierSummation is the current integrated volume of a given energy carrier measured on the inlet. The formatting and unit of measure for this value is specified in the EnergyCarrierUnitOfMeasure and EnergyCarrierSummationFormatting attributes (refer to Table 10-40).

The Energy consumption registered in CurrentSummationDelivered is not necessarily a direct function of this value. The quality of the energy carrier may vary from day to day, e.g. Gas may have different quality.

For heat and cooling meters the energy carrier is water at high or low temperature, the energy withdrawn from such a system is a function of the flow and the inlet and outlet temperature.

10.4.2.2.1.23 *CurrentOutletEnergyCarrierSummation* Attribute

CurrentOutletEnergyCarrierSummation is the current integrated volume of a given energy carrier measured on the outlet. The formatting and unit of measure for this value is specified in the EnergyCarrierUnitOfMeasure and EnergyCarrierSummationFormatting attributes (refer to Table 10-40).

10.4.2.2.1.24 *InletTemperature* Attribute

InletTemperature is the temperature measured on the energy carrier inlet.

The formatting and unit of measure for this value is specified in the TemperatureUnitOfMeasure and TemperatureFormatting attributes (refer to Table 10-40).

10.4.2.2.1.25 *OutletTemperature* Attribute

OutletTemperature is the temperature measured on the energy carrier outlet.

The formatting and unit of measure for this value is specified in the TemperatureUnitOfMeasure and TemperatureFormatting attributes (refer to Table 10-40).

10.4.2.2.1.26 *ControlTemperature* Attribute

ControlTemperature is a reference temperature measured on the meter used to validate the Inlet/Outlet temperatures.

The formatting and unit of measure for this value is specified in the TemperatureUnitOfMeasure and TemperatureFormatting attributes (refer to Table 10-40).

10.4.2.2.1.27 *CurrentInletEnergyCarrierDemand* Attribute

CurrentInletEnergyCarrierDemand is the current absolute demand on the energy carrier inlet.

The formatting and unit of measure for this value is specified in the EnergyCarrierUnitOfMeasure and EnergyCarrierDemandFormatting attributes (refer to Table 10-40).

For a heat or cooling meter this will be the current absolute flow rate measured on the inlet.

10.4.2.2.1.28 *CurrentOutletEnergyCarrierDemand* Attribute

CurrentOutletEnergyCarrierDemand is the current absolute demand on the energy carrier outlet.

The formatting and unit of measure for this value is specified in the EnergyCarrierUnitOfMeasure and EnergyCarrierDemandFormatting attributes (refer to Table 10-40).

For a heat or cooling meter this will be the current absolute flow rate measured on the outlet.

10.4.2.2.1.29 *PreviousBlockPeriodConsumptionDelivered* Attribute

The PreviousBlockPeriodConsumptionDelivered attribute represents the total value of Energy, Gas or Water delivered and consumed in the premises at the end of the previous Block Tariff Period. If supported, the PreviousBlockPeriodConsumptionDelivered attribute is updated at the end of each Block Tariff Period.

10.4.2.2.2 Summation TOU Information Set

The set of attributes shown in Table 10-33 provides a remote access to the Electric, Gas, or Water metering device's Time of Use (TOU) readings.

Note: TOU Information Attribute Set Attributes 0x0C-0x1D in this revision of this specification are provisionary and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

Table 10-33. TOU Information Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0100	*CurrentTier1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O

Id	Name	Type	Range	Acc	Def	M/O
0x0101	*CurrentTier1SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0102	*CurrentTier2SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0103	*CurrentTier2SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0104	*CurrentTier3SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFF	R	-	O
0x0105	*CurrentTier3SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0106	*CurrentTier4SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0107	*CurrentTier4SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0108	*CurrentTier5SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0109	*CurrentTier5SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x010A	*CurrentTier6SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x010B	*CurrentTier6SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x010C	*CurrentTier7SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x010D	*CurrentTier7SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x010E	*CurrentTier8SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x010F	*CurrentTier8SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0110	*CurrentTier9SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O

Id	Name	Type	Range	Acc	Def	M/O
0x0111	*CurrentTier9SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0112	*CurrentTier10SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0113	*CurrentTier10SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0114	*CurrentTier11SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0115	*CurrentTier11SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0116	*CurrentTier12SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0117	*CurrentTier12SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0118	*CurrentTier13SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0119	*CurrentTier13SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x011A	*CurrentTier14SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x011B	*CurrentTier1SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x011C	*CurrentTier15SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x011D	*CurrentTier15SummationReceived*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O

10.4.2.2.2.1 *CurrentTierNSummationDelivered* Attributes

Attributes CurrentTier1SummationDelivered through CurrentTierNSummationDelivered represent the most recent summed value of Energy, Gas, or Water delivered to the premises (i.e. delivered to the customer from the utility) at a specific price tier as defined by a TOU schedule or a real time pricing period. If optionally provided, attributes CurrentTier1SummationDelivered through CurrentTierNSummationDelivered are updated continuously as new measurements are made.

10.4.2.2.2.2 *CurrentTierNSummationReceived* Attributes

Attributes CurrentTier1SummationReceived through CurrentTierNSummationReceived represent the most recent summed value of Energy, Gas, or Water provided by the premises (i.e. received by the utility from the customer) at a specific price tier as defined by a TOU schedule or a real time pricing period. If optionally provided, attributes CurrentTier1SummationReceived through CurrentTierNSummationReceived are updated continuously as new measurements are made.

10.4.2.2.3 Meter Status Attribute Set

The Meter Status Attribute Set is defined in Table 10-34.

Table 10-34. Meter Status Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0200	*Status*	map8	0x00 to 0xFF	R	0x00	M
0x0201	*RemainingBatteryLife*	uint8	0x00 to 0xFF	R	-	O
0x0202	*HoursInOperation*	uint24	0x000000 to 0xFFFFFF	R	-	M:Heat M:Cooling O:others
0x0203	*HoursInFault*	uint24	0x000000 to 0xFFFFFF	R	-	O

10.4.2.2.3.1 *Status* Attribute

The Status attribute provides indicators reflecting the current error conditions found by the metering device. This attribute is an 8-bit field where when an individual bit is set, an error or warning condition exists. The behavior causing the setting or resetting each bit is device specific. In other words, the application within the metering device will determine and control when these settings are either set or cleared. Depending on the commodity type, the bits of this attribute will take on different meaning. Table 10-35 through Table 10-38 show the bit mappings for the Status attribute for Electricity, Gas, Water and Heating/Cooling, respectively. A battery-operated meter will report any change in state of the Status when it wakes up via a ZCL report attributes command. The ESI is expected to make alarms available to upstream systems together with consumption data collected from the battery operated meter.

Table 10-35. Mapping of the *Status* Attribute (Electricity)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Service Disconnect Open	Leak Detect	Power Quality	Power Failure	Tamper Detect	Low Battery	Check Meter

The definitions of the Electricity Status bits are:

Service Disconnect Open: Set to true when the service have been disconnected to this premises.

Leak Detect: Set to true when a leak have been detected.

Power Quality: Set to true if a power quality event have been detected such as a low voltage, high voltage.

Power Failure: Set to true during a power outage.

Tamper Detect: Set to true if a tamper event has been detected.

Low Battery: Set to true when the battery needs maintenance.

Check Meter: Set to true when a non fatal problem has been detected on the meter such as a measurement error, memory error, and self check error.

Table 10-36. Meter *Status* Attribute (Gas)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reverse Flow	Service Disconnect	Leak Detect	Low Pressure	Not Defined	Tamper Detect	Low Battery	Check Meter

The definitions of the Gas Status bits are:

Reverse Flow: Set to true if flow detected in the opposite direction to normal (from consumer to supplier).

Service Disconnect: Set to true when the service has been disconnected to this premises. Ex. The valve is in the closed position preventing delivery of gas.

Leak Detect: Set to true when a leak has been detected.

Low Pressure: Set to true when the pressure at the meter is below the meter's low pressure threshold value.

Tamper Detect: Set to true if a tamper event has been detected.

Low Battery: Set to true when the battery needs maintenance.

Check Meter: Set to true when a non fatal problem has been detected on the meter such as a measurement error, memory error, or self check error.

Table 10-37. Meter *Status* Attribute (Water)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reverse Flow	Service Disconnect	Leak Detect	Low Pressure	Pipe Empty	Tamper Detect	Low Battery	Check Meter

The definitions of the Water Status bits are:

Reverse Flow: Set to true if flow detected in the opposite direction to normal (from consumer to supplier).

Service Disconnect: Set to true when the service has been disconnected to this premises. Ex. The valve is in the closed position preventing delivery of water.

Leak Detect: Set to true when a leak has been detected.

Low Pressure: Set to true when the pressure at the meter is below the meter's low pressure threshold value.

Pipe Empty: Set to true when the service pipe at the meter is empty and there is no flow in either direction.

Tamper Detect: Set to true if a tamper event has been detected.

Low Battery: Set to true when the battery needs maintenance.

Check Meter: Set to true when a non fatal problem has been detected on the meter such as a measurement error, memory error, or self check error.

Table 10-38. Meter *Status* Attribute (Heat and Cooling)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Flow Sensor	Service Disconnect	Leak Detect	Burst Detect	Temperature Sensor	Tamper Detect	Low Battery	Check Meter

The definitions of the Heat and Cooling Status bits are:

Flow Sensor: Set to true when an error is detected on a flow sensor at this premises.

Service Disconnect: Set to true when the service has been disconnected to this premises. Ex. The valve is in the closed position preventing delivery of heat or cooling.

Leak Detect: Set to true when a leak has been detected.

Burst Detect: Set to true when a burst is detected on pipes at this premises.

Temperature Sensor: Set to true when an error is detected on a temperature sensor at this premises.

Tamper Detect: Set to true if a tamper event has been detected.

Low Battery: Set to true when the battery needs maintenance.

Check Meter: Set to true when a non fatal problem has been detected on the meter such as a measurement error, memory error, or self check error.

Note: It is not necessary to set aside Bit 7 as an “Extension Bit” for future expansion. If extra status bits are required an Extended Meter Status attribute may be added to support additional status values.

10.4.2.2.3.2 *RemainingBatteryLife* Attribute

RemainingBatteryLife represents the estimated remaining life of the battery in % of capacity. A setting of 0xFF indicates this feature is disabled. The range 0 - 100 where 100 = 100%, 0xFF = Unknown.

10.4.2.2.3.3 *HoursInOperation* Attribute

HoursInOperation is a counter that increments once every hour during operation. This may be used as a check for tampering.

Note: For meters that are not electricity meters turning off the meter does not necessarily prevent delivery of energy—but the meter might not be able to measure it.

10.4.2.2.3.4 *HoursInFault* Attribute

HoursInFault is a counter that increments once every hour when the device is in operation with a fault detected. This may be used as a check for tampering.

Note: For meters that are not electricity meters turning off the meter does not necessarily prevent delivery of energy—but the meter might not be able to measure it.

10.4.2.2.4 Formatting

The following set of attributes provides the ratios and formatting hints required to transform the received summations, consumptions, temperatures, or demands/ rates into displayable values. If the Multiplier and Divisor attribute values are non-zero, they are used in conjunction with the SummationFormatting, ConsumptionFormatting, DemandFormatting, and TemperatureFormatting attributes.

Equations required to accomplish this task are defined below:

Summation = Summation received * Multiplier / Divisor
(formatted using SummationFormatting)

Consumption = Consumption received * Multiplier / Divisor
(formatted using ConsumptionFormatting)

Demand = Demand received * Multiplier / Divisor
(formatted using DemandFormatting)

Temperature = Temperature received * Multiplier / Divisor

If the Multiplier and Divisor attribute values are zero, just the formatting hints defined in SummationFormatting, ConsumptionFormatting, DemandFormatting and TemperatureFormatting attributes are used.

The summation received, consumption received, demand received, and temperature received variables used above can be replaced by any of the attributes listed in sub-clauses 10.4.2.2.4.4, 10.4.2.2.4.5, 10.4.2.2.4.6, 10.4.2.2.4.11, 10.4.2.2.4.12, and 10.4.2.2.4.14.

Table 10-39 shows examples that demonstrate the relation between these attributes.

Table 10-39. Formatting Examples

Attribute	Example 1	Example 2	Example 3
Value as transmitted and received	52003	617	23629
Unit of Measure	kWh	CCF	kWh
Multiplier	1	2	6
Divisor	1000	100	10000
Number of Digits to the left of the Decimal Point	5	4	5
Number of Digits to the right of the Decimal Point	0	2	3
Suppress leading zeros	False	False	True
Displayed value	00052	0012.34	14.177

The Consumption Formatting Attribute Set is defined in Table 10-40.

Note: Consumption Formatting Attribute 0x07 in this revision of this specification is provisionary and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

Table 10-40. Formatting Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0300	*UnitofMeasure*	enum8	0x00 to0xFF	R	0x00	M
0x0301	*Multiplier*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x0302	*Divisor*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x0303	*SummationFormatting*	map8	0x00 to 0xFF	R	-	M
0x0304	*DemandFormatting*	map8	0x00 to 0xFF	R	-	O
0x0305	*HistoricalConsumptionFormatting*	map8	0x00 to 0xFF	R	-	O
0x0306	*MeteringDeviceType*	map8	0x00 to 0xFF	R	-	M
0x0307	*SiteID*	octstr	1 to 33 octets	R	-	O

Id	Name	Type	Range	Acc	Def	M/O
0x0308	*MeterSerialNumber*	octstr	1 to 25 octets	R	-	O
0x0309	*EnergyCarrierUnitOfMeasure*	enum8	0x00 to 0xFF	R	-	M*
0x030A	*EnergyCarrierSummationFormatting*	map8	0x00 to 0xFF	R	-	M*
0x030B	*EnergyCarrierDemandFormatting*	map8	0x00 to 0xFF	R	-	O
0x030C	*TemperatureUnitOfMeasure*	enum8	0x00 to 0xFF	R	-	M*
0x030D	*TemperatureFormatting*	map8	0x00 to 0xFF	R	-	M*

** Mandatory for Heat or Cooling; Optional for others*

10.4.2.2.4.1 *UnitofMeasure* Attribute

UnitofMeasure provides a label for the Energy, Gas, or Water being measured by the metering device. The units of measure apply to all summations, consumptions/ profile interval and demand/rate supported by this cluster. Other measurements such as the power factor are self describing. This attribute is an 8-bit enumerated field. The bit descriptions for this Attribute are listed in Table 10-41.

Table 10-41. *UnitofMeasure* Attribute Enumerations

Values	Description
0x00	kWh (Kilowatt Hours) & kW (Kilowatts) in pure binary format
0x01	m^3 (Cubic Meter) & m^3/h (Cubic Meter per Hour) in pure binary format
0x02	ft^3 (Cubic Feet) & ft^3/h (Cubic Feet per Hour) in pure binary format
0x03	ccf ((100 or Centum) Cubic Feet) & ccf/h ((100 or Centum) Cubic Feet per Hour) in pure binary format
0x04	US gl (US Gallons) & US gl/h (US Gallons per Hour) in pure binary format.
0x05	IMP gl (Imperial Gallons) & IMP gl/h (Imperial Gallons per Hour) in pure binary format
0x06	BTUs & BTU/h in pure binary format
0x07	Liters & l/h (Liters per Hour) in pure binary format
0x08	kPA (gauge) in pure binary format
0x09	kPA (absolute) in pure binary format
0x0A	mcf (1000 Cubic Feet) & mcf/h (1000 Cubic feet per hour) in pure binary format
0x0B	Unitless in pure binary format
0x0C	MJ (Mega Joule) and MJ/s (Mega Joule per second (MW)) in pure binary format
0x80	kWh (Kilowatt Hours) & kW (Kilowatts) in BCD format
0x81	m^3 (Cubic Meter) & m^3/h (Cubic Meter per Hour) in BCD format
0x82	ft^3 (Cubic Feet) & ft^3/h (Cubic Feet per Hour) in BCD format

Values	Description
0x83	ccf ((100 or Centum) Cubic Feet) & ccf/h ((100 or Centum) Cubic Feet per Hour) in BCD format
0x84	US gl (US Gallons) & US gl/h (US Gallons per Hour) in BCD format
0x85	IMP gl (Imperial Gallons) & IMP gl/h (Imperial Gallons per Hour) in BCD format
0x86	BTUs & BTU/h in BCD format
0x87	Liters & l/h (Liters per Hour) in BCD format
0x88	kPA (gauge) in BCD format
0x89	kPA (absolute) in BCD format
0x8A	mcf (1000 Cubic Feet) & mcf/h (1000 Cubic Feet per Hour) in BCD format
0x8B	unitless in BCD format
0x8C	MJ (Mega Joule) and MJ/s (Mega Joule per second (MW)) in BCD format

Note: When using BCD for meter reads, the values A to F are special values or indicators denoting “Opens”, “Shorts”, and etc. conditions when reading meter register hardware. Any SE device displaying the BCD based values to end users should use a non-decimal value to replace the A to F. In other words, a device could use an “*” in place of the special values or indicators.

10.4.2.2.4.2 *Multiplier* Attribute

Multiplier provides a value to be multiplied against a raw or uncompensated sensor count of Energy, Gas, or Water being measured by the metering device. If present, this attribute must be applied against all summation, consumption and demand values to derive the delivered and received values expressed in the unit of measure specified. This attribute must be used in conjunction with the Divisor attribute.

10.4.2.2.4.3 *Divisor* Attribute

Divisor provides a value to divide the results of applying the Multiplier Attribute against a raw or uncompensated sensor count of Energy, Gas, or Water being measured by the metering device. If present, this attribute must be applied against all summation, consumption and demand values to derive the delivered and received values expressed in the unit of measure specified. This attribute must be used in conjunction with the Multiplier attribute.

10.4.2.2.4.4 *SummationFormatting* Attribute

SummationFormatting provides a method to properly decipher the number of digits and the decimal location of the values found in the Summation Information Set of attributes. This attribute is to be decoded as follows:

Bits 0 to 2: Number of Digits to the right of the Decimal Point.

Bits 3 to 6: Number of Digits to the left of the Decimal Point.

Bit 7: If set, suppress leading zeros.

This attribute shall be used against the following attributes:

- CurrentSummationDelivered
- CurrentSummationReceived
- TOU Information attributes
- DFTSummation
- Block Information attributes

10.4.2.2.4.5 *DemandFormatting* Attribute

DemandFormatting provides a method to properly decipher the number of digits and the decimal location of the values found in the Demand-related attributes. This attribute is to be decoded as follows:

Bits 0 to 2: Number of Digits to the right of the Decimal Point.

Bits 3 to 6: Number of Digits to the left of the Decimal Point.

Bit 7: If set, suppress leading zeros.

This attribute shall be used against the following attributes:

- CurrentMaxDemandDelivered
- CurrentMaxDemandReceived
- InstantaneousDemand

10.4.2.2.4.6 *HistoricalConsumptionFormatting* Attribute

HistoricalConsumptionFormatting provides a method to properly decipher the number of digits and the decimal location of the values found in the Historical Consumption Set of attributes. This attribute is to be decoded as follows:

Bits 0 to 2: Number of Digits to the right of the Decimal Point.

Bits 3 to 6: Number of Digits to the left of the Decimal Point.

Bit 7: If set, suppress leading zeros.

This attribute shall be used against the following attributes:

- CurrentDayConsumptionDelivered
- CurrentDayConsumptionReceived
- PreviousDayConsumptionDelivered
- PreviousDayConsumptionReceived
- CurrentPartialProfileIntervalValue
- Intervals
- DailyConsumptionTarget

10.4.2.2.4.7 MeteringDeviceType Attribute

MeteringDeviceType provides a label for identifying the type of metering device present. The attribute are values representing Energy, Gas, Water, Thermal, Heat, Cooling, and mirrored metering devices. The defined values are represented in Table 10-42. (Note that these values represent an Enumeration, and not an 8-bit bitmap as indicated in the attribute description. For backwards compatibility reasons, the data type has not been changed, though the data itself should be treated like an enum.)

Where a mirror is provided for a battery-powered metering device, the mirror shall assume the relevant 'Mirrored Metering' device type (128-133) whilst the meter itself shall utilize the 'Metering' device type (1 to 6). It shall be the responsibility of the device providing the mirror to modify the Device Type shown on the mirror to that of a 'Mirrored Metering' device.

Table 10-42. *MeteringDeviceType* Attribute

Values	Description
0	Electric Metering
1	Gas Metering

Values	Description
2	Water Metering
3	Thermal Metering (deprecated)
4	Pressure Metering
5	Heat Metering
6	Cooling Metering
128	Mirrored Gas Metering
129	Mirrored Water Metering
130	Mirrored Thermal Metering (deprecated)
131	Mirrored Pressure Metering
132	Mirrored Heat Metering
133	Mirrored Cooling Metering

Note: Heat and cooling meters are used for measurement and billing of heat (and cooling) delivered through liquid (water) based central heating systems. The consumers are typically billed by the kWh, calculated from the flow and the temperatures in and out.

10.4.2.2.4.8 *SiteID* Attribute

The SiteID is a ZCL Octet String field capable of storing a 32 character string (the first octet indicates length) encoded in UTF-8 format. The SiteID is a text string, known in the UK as the M-PAN number for electricity, MPRN for gas and 'Stand Point' in South Africa. These numbers specify the meter point location in a standardized way. The field is defined to accommodate the number of characters typically found in the UK and Europe (16 digits). Generally speaking the field is numeric but is defined for the possibility of an alpha-numeric format by specifying an octet string.

10.4.2.2.4.9 *MeterSerialNumber* Attribute

The MeterSerialNumber is a ZCL Octet String field capable of storing a 24 character string (the first octet indicates length) encoded in UTF-8 format. It is used to provide a unique identification of the metering device.

10.4.2.2.4.10 *EnergyCarrierUnitOfMeasure* Attribute

The EnergyCarrierUnitOfMeasure specifies the unit of measure that the EnergyCarrier is measured in. This unit of measure is typically a unit of volume or flow and cannot be an amount of energy. The enumeration of this attribute is otherwise identical to the UnitofMeasure attribute (Table 10-41).

10.4.2.2.4.11 EnergyCarrierSummationFormatting Attribute

EnergyCarrierSummationFormatting provides a method to properly decipher the number of digits and the decimal location of the values found in the Summation- related attributes.

This attribute is to be decoded as follows:

Bits 0 to 2: Number of Digits to the right of the Decimal Point.

Bits 3 to 6: Number of Digits to the left of the Decimal Point.

Bit 7: If set, suppress leading zeros.

This attribute shall be used in relation with the following attributes:

- CurrentInletEnergyCarrierSummation

- CurrentOutletEnergyCarrierSummation

10.4.2.2.4.12 *EnergyCarrierDemandFormatting* Attribute

EnergyCarrierDemandFormatting provides a method to properly decipher the number of digits and the decimal location of the values found in the Demand-related attributes.

This attribute is to be decoded as follows:

Bits 0 to 2: Number of Digits to the right of the Decimal Point.

Bits 3 to 6: Number of Digits to the left of the Decimal Point.

Bit 7: If set, suppress leading zeros.

This attribute shall be used in relation with the following attributes:

- CurrentInletEnergyCarrierDemand
- CurrentOutletEnergyCarrierDemand
- CurrentDayMaxEnergyCarrierDemand
- PreviousDayMaxEnergyCarrierDemand
- CurrentMonthMaxEnergyCarrierDemand
- CurrentMonthMinEnergyCarrierDemand
- CurrentYearMinEnergyCarrierDemand
- CurrentYearMaxEnergyCarrierDemand

10.4.2.2.4.13 *TemperatureUnitOfMeasure* Attribute

The TemperatureUnitOfMeasure specifies the unit of measure that temperatures are measured in. The enumeration of this attribute is shown in Table 10-43.

Table 10-43. *TemperatureUnitOfMeasure* Enumeration

Values	Description
0x00	K (Degrees Kelvin) in pure Binary format.
0x01	°C (Degrees Celsius) in pure Binary format.
0x02	°F (Degrees Fahrenheit) in pure Binary format.
0x80	K (Degrees Kelvin) in BCD format.
0x81	°C (Degrees Celsius) in BCD format.
0x82	°F (Degrees Fahrenheit) in BCD format.

10.4.2.2.4.14 *TemperatureFormatting* Attribute

TemperatureFormatting provides a method to properly decipher the number of digits and the decimal location of the values found in the Temperature-related attributes. This attribute is to be decoded as follows:

Bits 0 to 2: Number of Digits to the right of the Decimal Point.

Bits 3 to 6: Number of Digits to the left of the Decimal Point.

Bit 7: If set, suppress leading zeros.

This attribute shall be used in relation with the following attributes:

- InletTemperature

- OutletTemperature
- ControlTemperature

10.4.2.2.5 Historical Consumption Attribute

The Historical Attribute Set is defined in Table 10-44.

Note: Historical Consumption Attributes 0x09-0x0E, 0x11 and 0x12 in this revision of this specification are provisional and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

Table 10-44. Historical Consumption Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0400	*InstantaneousDemand*	int24	-8,388,607 to 8,388,607	R	0	O
0x0401	*CurrentDayConsumptionDelivered*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x0402	*CurrentDayConsumptionReceived*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x0403	*PreviousDayConsumptionDelivered*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x0404	*PreviousDayConsumptionReceived*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x0405	*CurrentPartialProfileIntervalStartTimeDelivered*	UTC		R	-	O
0x0406	*CurrentPartialProfileIntervalStartTimeReceived*	UTC	0x000000 to 0xFFFFFF	R	-	O
0x0407	*CurrentPartialProfileIntervalValueDelivered*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x0408	*CurrentPartialProfileIntervalValueReceived*	uint24	0x000000 to 0xFFFFFF	R	-	O
0x0409	*CurrentDayMaxPressure*	uint48	0x000000 000000 to 0xFFFFFF FFFFFF	R	-	O
0x040a	*CurrentDayMinPressure*	uint48	0x000000 000000 to 0xFFFFFF FFFFFF	R	-	O
0x040b	*PreviousDayMaxPressure*	uint48	0x000000 000000 to 0xFFFFFF FFFFFF	R	-	O
0x040c	*PreviousDayMinPressure*	uint48	0x000000 000000 to 0xFFFFFF FFFFFF	R	-	O
0x040d	*CurrentDayMaxDemand*	int24	-8,388,607 to 8,388,607	R	-	O
0x040e	*PreviousDayMaxDemand*	int24	-8,388,607 to 8,388,607	R	-	O
0x040f	*CurrentMonthMaxDemand*	int24	-8,388,607 to 8,388,607	R	-	O
0x0410	*CurrentYearMaxDemand*	int24	-8,388,607 to 8,388,607	R	-	O
0x0411	*CurrentDayMaxEnergyCarrierDemand*	int24	-8,388,607 to 8,388,607	R	-	O
0x0412	*PreviousDayMaxEnergyCarrierDemand*	int24	-8,388,607 to 8,388,607	R	-	O
0x0413	*CurrentMonthMaxEnergyCarrierDemand*	int24	-8,388,607 to 8,388,607	R	-	O
0x0414	*CurrentMonthMinEnergyCarrierDemand*	int24	-8,388,607 to 8,388,607	R	-	O
0x0415	CurrentYearMaxEnergyCarrierDemand	int24	-8,388,607 to 8,388,607	R	-	O
0x0416	*CurrentYearMinEnergyCarrierDemand*	int24	-8,388,607 to 8,388,607	R	-	O

10.4.2.2.5.1 *InstantaneousDemand* Attribute

InstantaneousDemand represents the current Demand of Energy, Gas, or Water delivered or received at the premises. Positive values indicate demand delivered to the premises where negative values indicate demand received from the premises. InstantaneousDemand is updated continuously as new measurements are made. The frequency of updates to this field is specific to the metering device, but should be within the range of once every second to once every 5 seconds.

10.4.2.2.5.2 *CurrentDayConsumptionDelivered* Attribute

CurrentDayConsumptionDelivered represents the summed value of Energy, Gas, or Water generated and delivered to the premises since midnight local time. If optionally provided, CurrentDayConsumptionDelivered is updated continuously as new measurements are made.

10.4.2.2.5.3 *CurrentDayConsumptionReceived* Attribute

CurrentDayConsumptionReceived represents the summed value of Energy, Gas, or Water generated and received from the premises since midnight local time. If optionally provided, CurrentDayConsumptionReceived is updated continuously as new measurements are made.

10.4.2.2.5.4 *PreviousDayConsumptionDelivered* Attribute

PreviousDayConsumptionDelivered represents the summed value of Energy, Gas, or Water generated and delivered to the premises within the previous 24 hour period starting at midnight local time. If optionally provided, CurrentDayConsumptionDelivered is updated every midnight local time.

10.4.2.2.5.5 *PreviousDayConsumptionReceived* Attribute

PreviousDayConsumptionReceived represents the summed value of Energy, Gas, or Water generated and received from the premises within the previous 24 hour period starting at midnight local time. If optionally provided, CurrentDayConsumptionReceived is updated is updated every midnight local time.

10.4.2.2.5.6 *CurrentPartialProfileIntervalStartTimeDelivered* Attribute

CurrentPartialProfileIntervalStartTimeDelivered represents the start time of the current Load Profile interval being accumulated for commodity delivered.

10.4.2.2.5.7 *CurrentPartialProfileIntervalStartTimeReceived* Attribute

CurrentPartialProfileIntervalStartTimeReceived represents the start time of the current Load Profile interval being accumulated for commodity received.

10.4.2.2.5.8 *CurrentPartialProfileIntervalValueDelivered* Attribute

CurrentPartialProfileIntervalValueDelivered represents the value of the current Load Profile interval being accumulated for commodity delivered.

10.4.2.2.5.9 *CurrentPartialProfileIntervalValueReceived* Attribute

CurrentPartialProfileIntervalValueReceived represents the value of the current Load Profile interval being accumulated for commodity received.

10.4.2.2.5.10 *CurrentDayMaxPressure* Attribute

CurrentDayMaxPressure is the maximum pressure reported during a day from the water or gas meter.

10.4.2.2.5.11 *PreviousDayMaxPressure* Attribute

PreviousDayMaxPressure represents the maximum pressure reported during previous day from the water or gas meter.

10.4.2.2.5.12 *CurrentDayMinPressure* Attribute

CurrentDayMinPressure is the minimum pressure reported during a day from the water or gas meter.

10.4.2.2.5.13 *PreviousDayMinPressure* Attribute

PreviousDayMinPressure represents the minimum pressure reported during previous day from the water or gas meter.

10.4.2.2.5.14 *CurrentDayMaxDemand* Attribute

CurrentDayMaxDemand represents the maximum demand or rate of delivered value of Energy, Gas, or Water being utilized at the premises.

10.4.2.2.5.15 *PreviousDayMaxDemand* Attribute

PreviousDayMaxDemand represents the maximum demand or rate of delivered value of Energy, Gas, or Water being utilized at the premises.

Note: At the end of a day the metering device will transfer the CurrentDayMaxPressure into PreviousDayMaxPressure, CurrentDayMinPressure into PreviousDayMinPressure and CurrentDayMaxDemand into PreviousDayMaxDemand.

10.4.2.2.5.16 *CurrentMonthMaxDemand* Attribute

CurrentMonthMaxDemand is the maximum demand reported during a month from the meter.

For electricity, heat and cooling meters this is the maximum power reported in a month.

10.4.2.2.5.17 *CurrentYearMaxDemand* Attribute

CurrentYearMaxDemand is the maximum demand reported during a year from the meter.

For electricity, heat and cooling meters this is the maximum power reported in a year.

10.4.2.2.5.18 *CurrentDayMaxEnergyCarrierDemand* Attribute

CurrentDayMaxEnergyCarrierDemand is the maximum energy carrier demand reported during a day from the meter.

Note: At the end of a day the meter will transfer the CurrentDayMaxEnergyCarrierDemand into PreviousDayMaxEnergyCarrierDemand.

For heat and cooling meters this is the maximum flow rate on the inlet reported in a day.

10.4.2.2.5.19 *PreviousDayMaxEnergyCarrierDemand* Attribute

PreviousDayMaxEnergyCarrierDemand is the maximum energy carrier demand reported during the previous day from the meter.

10.4.2.2.5.20 *CurrentMonthMaxEnergyCarrierDemand* Attribute

CurrentMonthMaxEnergyCarrierDemand is the maximum energy carrier demand reported during a month from the meter.

For heat and cooling meters this is the maximum flow rate on the inlet reported in a month.

10.4.2.2.5.21 *CurrentMonthMinEnergyCarrierDemand* Attribute

CurrentMonthMinEnergyCarrierDemand is the minimum energy carrier demand reported during a month from the meter.

For heat and cooling meters this is the minimum flow rate on the inlet reported in a month.

Note: This attribute may be used to detect leaks if there has been no flow rate of zero in the last month.

10.4.2.2.5.22 *CurrentYearMaxEnergyCarrierDemand* Attribute

CurrentYearMaxEnergyCarrierDemand is the maximum energy carrier demand reported during a year from the meter.

For heat and cooling meters this is the maximum flow rate on the inlet reported in a year.

10.4.2.2.5.23 *CurrentYearMinEnergyCarrierDemand* Attribute

CurrentYearMinEnergyCarrierDemand is the minimum energy carrier demand reported during a year from the heat meter.

For heat and cooling meters this is the minimum flow rate on the inlet reported in a year.

Note: This attribute may be used to detect leaks if there has been no flow rate of zero in the last year

10.4.2.2.6 Load Profile Configuration

The Load Profile Configuration Attribute Set is defined in Table 10-45.

Table 10-45. Load Profile Configuration Attribute Set

Identifier	Name	Type	Range	Acc	Default	M/O
0x0500	*MaxNumberOfPeriodsDelivered*	uint8	0x00 to 0xFF	R	0x18	O

10.4.2.2.6.1 *MaxNumberOfPeriodsDelivered* Attribute

MaxNumberofPeriodsDelivered represents the maximum number of intervals the device is capable of returning in one Get Profile Response command. It is required MaxNumberofPeriodsDelivered fit within the default Fragmentation ASDU size of 128 bytes, or an optionally agreed upon larger Fragmentation ASDU size supported by both devices. Please refer to [Z1] for further details on Fragmentation settings.

10.4.2.2.7 Supply Limit Attributes

This set of attributes is used to implement a “Supply Capacity Limit” program where the demand at the premises is limited to a preset consumption level over a preset period of time. Should this preset limit be exceeded the meter could interrupt supply to the premises or to devices within the premises. The supply limit information in this attribute set can be used by In-Home displays, PCTs, or other devices to display a warning when the supply limit is being approached. The Supply Limit Attribute Set is defined in Table 10-46.

Table 10-46. Supply Limit Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0600	*CurrentDemandDelivered*	uint24	0x000000 to 0xFFFFFF	R		O
0x0601	*DemandLimit*	uint24	0x000000 to 0xFFFFFF	R		O
0x0602	*DemandIntegrationPeriod*	uint8	0x01 to 0xFF	R	-	O
0x0603	*NumberOfDemandSubintervals*	uint8	0x01 to 0xFF	R	-	O

10.4.2.2.7.1 *CurrentDemandDelivered* Attribute

CurrentDemandDelivered represents the current Demand of Energy, Gas, or Water delivered at the premises. CurrentDemandDelivered may be continuously updated as new measurements are acquired, but at a minimum CurrentDemandDelivered must be updated at the end of each integration sub- period, which can be obtained by dividing the DemandIntegrationPeriod by the NumberOfDemandSubintervals.

This attribute shall be adjusted using the Multiplier and Divisor attributes found in the Formatting Attribute Set and can be formatted using the DemandFormatting attribute. The final result represents an engineering value in the unit defined by the UnitofMeasure attribute.

10.4.2.2.7.2 *DemandLimit* Attribute

DemandLimit reflects the current supply demand limit set in the meter. This value can be compared to the CurrentDemandDelivered attribute to understand if limits are being approached or exceeded.

Adjustment and formatting of this attribute follow the same rules as the CurrentDemandDelivered.

A value of "0xFFFFFF" indicates "demand limiting" is switched off.

10.4.2.2.7.3 *DemandIntegrationPeriod* Attribute

DemandIntegrationPeriod is the number of minutes over which the CurrentDemandDelivered attribute is calculated. Valid range is 0x01 to 0xFF. 0x00 is a reserved value.

10.4.2.2.7.4 *NumberOfDemandSubintervals* Attribute

NumberOfDemandSubintervals represents the number of subintervals used within the DemandIntegrationPeriod. The subinterval duration (in minutes) is obtained by dividing the DemandIntegrationPeriod by the NumberOfDemandSubintervals. The CurrentDemandDelivered attribute is updated at each subinterval. Valid range is 0x01 to 0xFF. 0x00 is a reserved value.

As a Rolling Demand example, DemandIntegrationPeriod could be set at 30 (for 30 minute period) and NumberOfDemandSubintervals could be set for 6. This would provide 5 minute (30/6 = 5) subinterval periods.

As a Block Demand example, DemandIntegrationPeriod could be set at 30 (for 30 minute period) and NumberOfDemandSubintervals could be set for 1. This would provide a single 30 minute subinterval period.

10.4.2.2.8 Block Information Set

The set of attributes shown in Table 10-47 provides a remote access to the Electric, Gas, or Water metering device's block readings. The Block Information attribute set supports Block pricing and combined Tier-Block pricing, the number of blocks is one greater than the number of block thresholds defined in the Pricing cluster.

Table 10-47. Block Information Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0700	*CurrentNoTierBlock1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0701	*CurrentNoTierBlock2SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0702	*CurrentNoTierBlock3SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x070*N*	*CurrentNoTierBlock*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x070f	*CurrentNoTierBlock16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0710	*CurrentTier1Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O

Id	Name	Type	Range	Acc	Def	M/O
0x0711	*CurrentTier1Block2SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0712	*CurrentTier1Block3SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x071*N*	*CurrentTier1Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x071f	*CurrentTier1Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0720	*CurrentTier2Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x072*N*	*CurrentTier2Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x072f	*CurrentTier2Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0730	*CurrentTier3Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x073*N*	*CurrentTier3Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x073f	*CurrentTier3Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0740	*CurrentTier4Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x074*N*	*CurrentTier4Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x074f	*CurrentTier4Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0750	*CurrentTier5Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x075*N*	*CurrentTier5Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x075f	*CurrentTier5Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0760	*CurrentTier6Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x076*N*	*CurrentTier6Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O

Id	Name	Type	Range	Acc	Def	M/O
0x076f	*CurrentTier6Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0770	*CurrentTier7Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x077*N*	*CurrentTier7Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x077f	*CurrentTier7Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0780	*CurrentTier8Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x078*N*	*CurrentTier8Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x078f	*CurrentTier8Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x0790	*CurrentTier9Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x079*N*	*CurrentTier9Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x079f	*CurrentTier9Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x007a0	*CurrentTier10Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07a*N*	*CurrentTier10Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07af	*CurrentTier10Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07b0	*CurrentTier11Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07b*N*	*CurrentTier11Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07bf	*CurrentTier11Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O

Id	Name	Type	Range	Acc	Def	M/O
0x07C0	*CurrentTier12Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07c*N*	*CurrentTier12Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07cf	*CurrentTier12Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07d0	*CurrentTier13Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07d*N*	*CurrentTier13Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07df	*CurrentTier13Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07e0	*CurrentTier14Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07e*N*	*CurrentTier14Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07ef	*CurrentTier14Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07f0	*CurrentTier15Block1SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07f*N*	*CurrenTier15Block*N+1*SummationDelivered…*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O
0x07ff	*CurrentTier15Block16SummationDelivered*	uint48	0x000000000000 to 0xFFFFFFFFFFFF	R	-	O

10.4.2.2.8.1 *CurrentTierNBlockNSummationDelivered* Attributes

Attributes CurrentNoTierBlock1SummationDelivered through CurrentTier15Block16SummationDelivered represent the most recent summed value of Energy, Gas, or Water delivered to the premises (i.e. delivered to the customer from the utility) at a specific price tier as defined by a TOU schedule, Block Threshold or a real time pricing period. If optionally provided, attributes CurrentNoTierBlock1SummationDelivered through CurrentTier15Block16SummationDelivered are updated continuously as new measurements are made.

Note: SummationFormatting shall be used against the Block Information attribute set. The expected practical limit for the number of Block attributes supported is 32. The CurrentTierNBlockNSummationDelivered attributes are reset at the start of each Block Threshold Period.

10.4.2.2.9 Alarms Set

The set of attributes shown in Table 10-48 provides a means to control which alarms may be generated from the meter.

Note: Alarms Attribute Set in this revision of this specification is provisional and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

Table 10-48. Alarm Attribute Set

Id	Name	Type	Range	Acc	Default	M/O
0x0800	*GenericAlarmMask*	map16	0x0000 - 0xffff	RW	0xffff	O
0x0801	*ElectricityAlarmMask*	map32	0x00000000 - 0xffffffff	RW	0xffffffff	O
0x0802	*Generic Flow/PressureAlarmMask*	map16	0x0000 - 0xffff	RW	0xffff	O
0x0803	*WaterSpecificAlarmMask*	map16	0x0000 - 0xffff	RW	0xffff	O
0x0804	*HeatandCoolingSpecificAlarmMask*	map16	0x0000 - 0xffff	RW	0xffff	O
0x0805	*GasSpecificAlarmMask*	map16	0x0000 - 0xffff	RW	0xffff	O

10.4.2.2.9.1 *AlarmMask* Attributes

The AlarmMask attributes of the Alarm Attribute Set specify whether each of the alarms listed in the corresponding alarm group in Table 10-49 through Table 10-55 is enabled. When the bit number corresponding to the alarm number (minus the group offset) is set to 1, the alarm is enabled, else it is disabled. Bits not corresponding to a code in the respective table are reserved.

10.4.2.2.9.2 Alarm Codes

The alarm codes are organized in logical groups corresponding to the meter type as listed in Table 10-49. The three main alarm groups are: Generic, Electricity, and Flow/ Pressure. The Flow/Pressure Alarm Group is further divided into Generic Flow/Pressure, Water Specific, Heat and Cooling Specific, and Gas Specific. It is left for the manufacturer to select which (if any) alarm codes to support.

Table 10-49. Alarm Code Groups

Alarm Code	Alarm Condition
00-0F	Generic Alarm Group
10-2F	Electricity Alarm Group
30-7F	Flow/Pressure Alarm Group
30-3F	Generic Flow/Pressure Alarm Group
40-4F	Water Specific Alarm Group
50-5F	Heat and Cooling Specific Alarm Group
60-6F	Gas Specific Alarm Group

The generic Alarm Group maps the status from the MeterStatus attribute into a corresponding alarm. Hence, depending on the meter type, an alarm belonging to the Generic Alarm Group may have a different meaning. See sub-clause 10.4.2.2.3. In the case of overlap of alarm codes from the Generic Alarm Group with codes in other groups, e.g. Burst Detect, it is recommended to only use the code of the Generic Alarm Group, as shown in Table 10-50.

Table 10-50. Generic Alarm Group

Alarm Code	Alarm Condition
00	Check Meter
01	Low Battery
02	Tamper Detect
03	Electricity: Power Failure Gas: Not Defined Water: Pipe Empty Heat/Cooling: Temperature Sensor
04	Electricity: Power Quality Gas: Low Pressure Water: Low Pressure Heat/Cooling: Burst Detect
05	Leak Detect
06	Service Disconnect
07	Electricity: Reserved Gas: Reverse Flow Water: Reverse Flow Heat/Cooling: Flow Sensor
08-0F	Reserved

The Electricity Alarm Group defines alarms specific for electricity meters as defined in Table 10-51.

Table 10-51. Electricity Alarm Group

Alarm Code	Alarm Condition
10	Low Voltage L1
11	High Voltage L1
12	Low Voltage L2
13	High Voltage L2
14	Low Voltage L3
15	High Voltage L3
16	Over Current L1
17	Over Current L2
18	Over Current L3
19	Frequency too Low L1

Alarm Code	Alarm Condition
1A	Frequency too High
1B	Frequency too Low L2
1C	Frequency too High
1D	Frequency too Low L3
1E	Frequency too High
1F	Ground Fault
20	Electric Tamper
21-2F	Reserved

The Generic Flow/Pressure Alarm Group defines alarms specific for Flow/Pressure based meters i.e. Water, Heat, Cooling, or Gas meters as defined in Table 10-52.

Table 10-52. Generic Flow/Pressure Alarm Group

Alarm Code	Alarm Condition
30	Burst detect
31	Pressure too low
32	Pressure too high
33	Flow sensor communication error
34	Flow sensor measurement fault
35	Flow sensor reverse flow
36	Flow sensor air detect
37	Pipe empty
38-3F	Reserved

The Water Specific Alarm Group defines alarms specific for Water meters as defined in Table 10-53.

Table 10-53. Water Specific Alarm Group

Alarm	Alarm Condition
40-4F	Reserved

The Heat and Cooling Specific Alarm Group defines alarms specific for Heat or Cooling meters as defined in Table 10-54.

Table 10-54. Heat and Cooling Specific Alarm Group

Alarm Code	Alarm Condition
50	Inlet Temperature Sensor Fault
51	Outlet Temperature Sensor
52-5F	Reserved

The Gas Specific Alarm Group defines alarms specific for Gas meters as defined in Table 10-55.

Table 10-55. Gas Specific Alarm Group

Alarm Code	Alarm Condition
60-6F	Reserved

10.4.2.3 Server Commands

10.4.2.3.1 Commands Generated

The command IDs generated by the Metering server cluster are listed in Table 10-56.

Table 10-56. Generated Command IDs for the Metering Server

Id	Name	M/O
0x00	*Get Profile Response*	O
0x01	*Request Mirror*	O
0x02	*Remove Mirror*	O
0x03	*Request Fast Poll Mode Response*	O

10.4.2.3.1.1 *Get Profile Response* Command

10.4.2.3.1.1.1 Payload Format

The Get Profile Response command payload shall be formatted as illustrated in Figure 10-30.

Figure 10-30. Format of the *Get Profile Response* Command Payload

Octets	4	1	1	1	Variable
Data Type	UTC	enum8	enum8	uint8	Series of uint24s
Field Name	EndTime	Status	ProfileInterval Period	NumberOfPeriodsDelivered	Intervals

10.4.2.3.1.1.2 Payload Details

EndTime: 32-bit value (in UTC) representing the end time of the most chronologically recent interval being requested. Example: Data collected from 2:00 PM to 3:00 PM would be specified as a 3:00 PM interval (end time). It is important to note that the current interval accumulating is not included in most recent block but can be retrieved using the CurrentPartialProfileIntervalValue attribute.

Status: Table 10-57 lists the valid values returned in the Status field.

Table 10-57. Status Field Values

Value	Description
0x00	Success
0x01	Undefined Interval Channel requested
0x02	Interval Channel not supported
0x03	Invalid End Time
0x04	More periods requested than can be returned
0x05	No intervals available for the requested time

ProfileIntervalPeriod: Represents the interval or time frame used to capture metered Energy, Gas, and Water consumption for profiling purposes. ProfileIntervalPeriod is an enumerated field representing the timeframes listed in Table 10-58.

Table 10-58. ProfileIntervalPeriod Timeframes

Enumerated Value	Timeframe
0	Daily
1	60 minutes
2	30 minutes
3	15 minutes
4	10 minutes
5	7.5 minutes
6	5 minutes
7	2.5 minutes

NumberofPeriodsDelivered: Represents the number of intervals the device is returning. Please note the number of periods returned in the Get Profile Response command can be calculated when the packets are received and can replace the usage of this field. The intent is to provide this information as a convenience.

Intervals: Series of interval data captured using the period specified by the ProfileIntervalPeriod field. The content of the interval data depend of the type of information requested using the Channel field in the Get Profile Command. Data is organized in a reverse chronological order, the most recent interval is transmitted first and the oldest interval is transmitted last. Invalid intervals should be marked as 0xFFFFFF.

10.4.2.3.1.1.3 When Generated

This command is generated when the Client command GetProfile is received. Please refer to sub-clause 10.4.2.4.1.1.

10.4.2.3.1.2 Request Mirror Command

This command is used to request the ESI to mirror Metering Device data.

10.4.2.3.1.2.1 Payload Details

There are no fields for this command.

10.4.2.3.1.2.2 Effect on Receipt

On receipt of this command, the Server shall send a RequestMirrorReponse command (see sub-clause 10.4.2.4.1.2).

10.4.2.3.1.3 Remove Mirror Command

This command is used to request the ESI to remove its mirror of Metering Device data. The device sending the Remove Mirror command to the ESI shall send the command to the mirror endpoint to be removed. Only the device that created the mirror on the ESI or the ESI itself should be allowed to remove the mirror from the ESI.

10.4.2.3.1.3.1 Payload Details

There are no fields for this command.

10.4.2.3.1.3.2 Effect on Receipt

On receipt of this command, the Server shall send a MirrorRemoved command (see sub-clause 10.4.2.4.1.3).

10.4.2.3.1.4 Request Fast Poll Mode Response Command

10.4.2.3.1.4.1 Payload Format

The Request Fast Poll Mode Response command payload shall be formatted as illustrated in Figure 10-31.

Figure 10-31. Format of the *Request Fast Poll Mode Response* Command Payload

Octets	1	4
Data Type	uint8	UTC
Field Name	Applied Update Period (seconds) (M)	Fast Poll Mode End Time (M)

10.4.2.3.1.4.2 Payload Details

Applied Update Period: The period at which metering data shall be updated. This may be different than the requested fast poll. If the Request Fast Poll Rate is less than Fast Poll Update Period Attribute, it shall use the Fast Poll Update Period Attribute. Otherwise, the Applied Update Period shall be greater than or equal to the minimum Fast Poll Update Period Attribute and less than or equal to the Requested Fast Poll Rate.

Fast Poll Mode End Time: UTC time that indicates when the metering server will terminate fast poll mode and resume updating at the rate specified by DefaultUpdatePeriod. For example, one or more metering clients may request fast poll mode while the metering server is already in fast poll mode. The intent is that the fast poll mode will not be extended since this scenario would make it possible to be in fast poll mode longer than 15 minutes.

10.4.2.3.1.4.3 When Generated

This command is generated when the client command Request Fast Poll Mode is received.

10.4.2.3.1.4.4 Effect on Receipt

On receipt of this command, the device may request or receive updates not to exceed the Applied Update Period until Fast Poll Mode End Time.

10.4.2.4 Client Commands

10.4.2.4.1 Commands Generated

The command IDs generated by the Metering client cluster are listed in Table 10-59.

Table 10-59. Generated Command IDs for the Metering Client

Command Identifier Field Value	Description	M/O
0x00	*Get Profile*	O
0x01	*Request Mirror Response*	O
0x02	*Mirror Removed*	O
0x03	*Request Fast Poll Mode*	O

10.4.2.4.1.1 *Get Profile* Command

The Get Profile command payload shall be formatted as illustrated in Figure 10-32.

Figure 10-32. Format of the *Get Profile* Command Payload

Octets	1	4	1
Data Type	enum8	UTC	uint8
Field Name	Interval Channel	End Time	NumberOfPeriods

10.4.2.4.1.1.1 Payload Details

Interval Channel: Enumerated value used to select the quantity of interest returned by the GetProfileReponse command. The Interval Channel values are listed in Table 10-60.

Table 10-60. Interval Channel Values

Enumerated Value	Description
0	Consumption
1	Consumption Received

EndTime: 32-bit value (in UTC) used to select an Intervals block from all the Intervals blocks available. The Intervals block returned is the most recent block with its EndTime equal or older to the one provided. The most recent Intervals block is requested using an End Time set to 0x00000000, subsequent Intervals block are requested using an End time set to the EndTime of the previous block - (number of intervals of the previous block * ProfileIntervalPeriod).

NumberofPeriods: Represents the number of intervals being requested. This value can’t exceed the size stipulated in the MaxNumberOfPeriodsDelivered attribute. If more intervals are requested than can be delivered, the GetProfileResponse will return the number of intervals equal to MaxNumberOfPeriodsDelivered. If fewer intervals are available for the time period, only those available are returned.

10.4.2.4.1.1.2 When Generated

The GetProfile command is generated when a client device wishes to retrieve a list of captured Energy, Gas or water consumption for profiling purposes. Due to the potentially large amount of profile data available, the client device should store previously gathered data and only request the most current data. When initially gathering significant amounts of historical interval data, the GetProfile command should not be issued any more frequently than 7.5 seconds to prevent overwhelming the ZigBee network.

10.4.2.4.1.1.3 Command Processing Response

If failure occurs in recognizing or processing the payload of the GetProfile command, the appropriate enumerated ZCL status (as defined in Chapter 2) will be returned. On success, a non-Default Response is returned without a ZCL status code.

10.4.2.4.1.1.4 Effect on Receipt

On receipt of this command, the device shall send a GetProfileReponse command (see sub-clause 10.4.2.3.1.1).

10.4.2.4.1.2 *Request Mirror Response* Command

The Request Mirror Response Command allows the ESI to inform a sleepy Metering Device it has the ability to store and mirror its data.

10.4.2.4.1.2.1 Payload Format

The Request Mirror Response command payload shall be formatted as illustrated in Figure 10-33.

Figure 10-33. Format of the *Request Mirror Response* Command Payload

Octets	2
Data Type	uint16
Field Name	EndPoint ID

10.4.2.4.1.2.2 Payload Details

EndPoint ID: 16 Bit Unsigned Integer indicating the End Point ID to contain the Metering Devices meter data. Valid End Point ID values are 0x0001 to 0x00F0. If the ESI is able to mirror the Metering Device data, the low byte of the unsigned 16 bit integer shall be used to contain the eight bit EndPoint ID. If the ESI is unable to mirror the Metering Device data, EndPoint ID shall be returned as 0xFFFF. All other EndPoint ID values are reserved. If valid, the Metering device shall use the EndPoint ID to forward its metered data.

10.4.2.4.1.3 *Mirror Removed* Command

The Mirror Removed Command allows the ESI to inform a sleepy Metering Device mirroring support has been removed or halted.

10.4.2.4.1.3.1 Payload Format

The Mirror Removed command payload shall be formatted as illustrated in Figure 10-34.

Figure 10-34. Format of the *Mirror Removed* Command Payload

Octets	2
Data Type	uint16
Field Name	Removed EndPoint ID

10.4.2.4.1.3.2 Payload Details

Removed EndPoint ID: 16 Bit Unsigned Integer indicating the End Point ID previously containing the Metering Device's meter data.

10.4.2.4.1.4 *Request Fast Poll Mode* Command

10.4.2.4.1.4.1 Payload Format

The Request Fast Poll Mode shall be formatted as illustrated in Figure 10-35.

Figure 10-35. Format of the *Request Fast Poll Mode* Command Payload

Octets	1	1
Data Type	uint8	uint8
Field Name	Fast Poll Update Period (seconds)	Duration (minutes)

10.4.2.4.1.4.2 Payload Details

Fast Poll Update Period: Desired fast poll period not to be less than the FastPollUpdatePeriod attribute.

Duration: Desired duration for the server to remain in fast poll mode not to exceed 15 minutes as specified in sub-clause 10.4.3.2.

10.4.2.4.1.4.3 When Generated

The Request Fast Poll Mode command is generated when the metering client wishes to receive near real-time updates of InstantaneousDemand. Fast poll mode shall only be requested as a result of user interaction (for example, the pushing of a button or activation of fast poll mode by a menu choice).

10.4.2.4.1.4.4 Effect on Receipt

The metering device may continuously update InstantaneousDemand as measurements are acquired, but at a minimum InstantaneousDemand must be updated at the end of each FastPollUpdatePeriod.

10.4.3 Metering Application Guidelines

10.4.3.1 Attribute Reporting

Attribute reporting may be used for sending information in the Reading Information, TOU Information, Meter Status, and Historical Consumption attribute sets. Use of the Report Attribute command without report configuration may be used for unsolicited notification of an attribute value change. Sleepy devices may have to poll.

10.4.3.2 Fast Polling or Reporting for Monitoring Energy Savings

Client devices, such as an energy gateway, smart thermostat, or in-home displays can monitor changes to energy saving settings within the premises and give users near real time feedback and results. The Metering cluster can support this by using Attribute Reporting and sending updates at a much faster rate for a short period of time. Client devices can also perform a series of Attribute reads to accomplish the same task. In either case, requests or updates shall be limited to a maximum rate of once every two seconds for a maximum period of 15 minutes. These limitations are required to ensure Smart Energy profile based devices do not waste available bandwidth or prevent other operations within the premises.

10.4.3.3 Metering Data Updates

The frequency and timeliness of updating metering data contained in the Metering Cluster attributes and Profile Intervals is up to the individual Metering device manufacturer's capabilities. As a best practice recommendation, updates of the metering data should not cause delivery of the information to end devices more often than once every 30 seconds. End devices should also not request information more often than once every 30 seconds.

The Fast Polling attributes and commands shall be used by client devices requesting information more often than once every 30 seconds.

10.4.3.3.1 Fast Polling Periods

Since the DefaultUpdatePeriod specifies the normal update interval and FastPollUpdatePeriod specifies the fastest possible update interval, it is recommended that metering clients read these attributes to determine the optimal normal/fast polling interval and the optimal fast poll period to request. Client devices shall not request data more frequent than FastPollUpdatePeriod or the AppliedUpdatePeriod.

10.4.3.4 Mirroring

SE Profile specifies Mirror support in the Metering cluster to store and provide access to data from metering devices on battery power. Devices with resources to support mirroring advertize the capability using the Basic Attribute Physical Environment.

10.4.3.4.1 Discovery

The SE standard does not prescribe how Mirroring is implemented. Devices may query the Basic Cluster attribute PhysicalEnvironment to determine Mirrored device capacity prior to CBKE (see sub-clause 10.4.3.4.2). This would allow a battery based end device to discover if an ESI has capacity to mirror data prior to the process of joining the network in a secure manner, thereby reducing retry attempts. This would also enhance the service discovery of the ZDO Match Descriptor that would be used to determine if an endpoint can request the setup and removal of a mirrored Metering cluster. Once a device has joined the network and performed CBKE, it can then request setup of a mirrored metering cluster. ZDO Discovery should be supported to allow HAN devices to discover the mirror endpoints; only active mirror endpoints shall be discoverable.

10.4.3.4.2 Mirror Attributes

The mandatory Basic, Metering, and (where applicable) Prepayment attributes shall be supported. The Basic Cluster PhysicalEnvironment attribute shall be supported on ESIs supporting mirroring functionality; an enumerated value of 0x01 would indicate that the device has the capacity to mirror an end device; a value of 0x00 would specify an “Unspecified environment” per the ZCL specification. Only the Basic cluster for devices capable of providing a mirror shall have the PhysicalEnvironment attribute set to 0x01. The ZCL Report Attribute command shall be used to push data to the mirror. Only the metering device that has been granted a mirror on a certain endpoint is allowed to push data to that endpoint. The ZCL Not Authorized return status shall be used to provide access control. The use of ZCL Report Configuration shall not be required to generate Report Attribute Command.

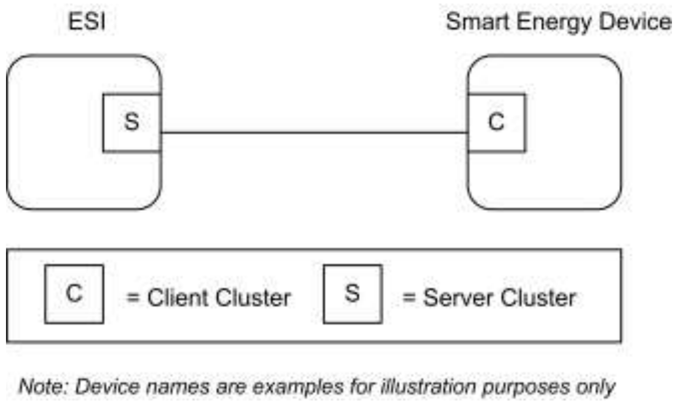
Manufacturers will design and manufacture devices to meet customer requirement specifications that will state the functionality of the battery powered meter and therefore devices supporting mirroring in the field will also have to support those requirements through an appropriate choice of optional attributes. Battery powered devices will report attributes to the mirror as required by the customer specification. In the event that the mirror is out of memory space or cannot support the attribute it shall respond ATTRIBUTE_UNSUPPORTED back to the battery-powered meter. The same response (ATTRIBUTE_UNSUPPORTED) will be sent to a device querying the mirror for an attribute it doesn't support. A device querying the mirror for an attribute that is supported but not yet available (the battery powered meter hasn't yet sent the attribute) shall receive a response ATTRIBUTE_UNAVAILABLE from the mirror.

10.5 Messaging

10.5.1 Overview

This cluster provides an interface for passing text messages between devices. Messages are expected to be delivered via the ESI and then unicast to all individually registered devices implementing the Messaging Cluster on the network, or just made available to all devices for later pickup. Nested and overlapping messages are not allowed. The current active message will be replaced if a new message is received by the ESI.

Figure 10-36. Messaging Cluster Client/Server Example


ESI
Smart Energy Device
S
C
C
= Client Cluster
S
= Server Cluster
Note: Device names are examples for illustration purposes only

Please note the ESI is defined as the Server due to its role in acting as the proxy for upstream message management systems and subsequent data stores.

10.5.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

10.5.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	SEMS	Type 1 (client to server)

10.5.1.3 Cluster Identifiers

Identifier	Name
0x0703	Messaging (Smart Energy)

10.5.2 Server

10.5.2.1 Dependencies

Support for ZCL Data Types.

Anonymous Inter-PAN transmission mechanism.

No dependencies exist for other Smart Energy Clusters.

10.5.2.2 Attributes

None

10.5.2.3 Commands Generated

The command IDs generated by the Messaging server cluster are listed in Table 10-61.

Table 10-61. Generated Command IDs for the Messaging Server

Command Identifier Field Value	Description	M/O
0x00	*Display Message*	M
0x01	*Cancel Message*	M

10.5.2.3.1 *Display Message* Command

10.5.2.3.1.1 Payload Format

The Display Message command payload shall be formatted as illustrated in Figure 10-37.

Figure 10-37. Format of the *Display Message* Command Payload

Octets	4	1	4	2	Variable
Data Type	uint32	map8	UTC	uint16	string
Field Name	Message ID	Message Control	Start Time	Duration In Minutes	Message

10.5.2.3.1.1.1 Payload Details

Message ID: A unique unsigned 32-bit number identifier for this message. It’s expected the value contained in this field is a unique number managed by upstream systems or a UTC based time stamp (UTC data type) identifying when the message was issued.

MessageControl: An 8-bit bitmap field indicating the need to optionally pass the message onto the Anonymous Inter-PAN transmission mechanism or that a user confirmation is required for a message. Bit encoding of this field is outlined in Table 10-62.

Table 10-62. Message Control Field Bit Map

Bits	Enumeration	Value	Description
0 to 1	Normal transmission only	0	Send message through normal command function to client.
	Normal and Anonymous Inter- PAN transmission	1	Send message through normal command function to client and pass message onto the Anonymous Inter-PAN transmission mechanism.

Bits	Enumeration	Value	Description
	Anonymous Inter- PAN transmission only	2	Send message through the Anonymous Inter- PAN transmission mechanism.
	Reserved	3	Reserved value for future use.
2 to 3	Low	0	Message to be transferred with a low level of importance.
	Medium	1	Message to be transferred with a medium level of importance.
	High	2	Message to be transferred with a high level of importance.
	Critical	3	Message to be transferred with a critical level of importance.
4 to 6	*Reserved*	*N/A*	These bits are reserved for future use.
7	Message Confirmation	0	Message Confirmation not required.
		1	Message Confirmation required.

If the Anonymous Inter-PAN transmission mechanism is not supported on a particular device, Bits 0 to 6 can be ignored.

The Message Confirmation bit indicates the message originator requests a confirmation of receipt from a Utility Customer. If confirmation is required, the device should display the message or alert the user until it is either confirmed via a button, by selecting a confirmation option on the device, or the message expires. Confirmation is typically used when the Utility is sending down information such as a disconnection notice, or prepaid billing information.

Note: It is desired that the device provide a visual indicator (flashing display or indicate with its LEDs as examples) that a message requiring confirmation is being displayed, and requires confirmation.

Start Time: A UTC field to denote the time at which the message becomes valid. A Start Time of 0x00000000 is a special time denoting “now.” If the device would send an event with a Start Time of now, adjust the Duration In Minutes field to correspond to the remainder of the event.

Duration In Minutes: An unsigned 16-bit field is used to denote the amount of time in minutes after the Start Time during which the message is displayed. A Maximum value of 0xFFFF means “until changed”.

Message: A ZCL String containing the message to be delivered. The String shall be encoded in the UTF-8 format. Please note: Since the Anonymous Inter-PAN transmission mechanism does not support fragmentation and is limited in its message size, any message forwarded will be truncated to match the maximum message length supported. For messages sent through the Anonymous Inter-PAN transmission mechanism and received by devices that display messages smaller than 80 bytes, they shall have the ability to receive up to an 80 byte message. Devices will have the ability to choose the methods for managing messages that are larger than can be displayed (truncation, scrolling, etc.).

For supporting larger messages sent over the SE Profile network, both devices must agree upon a common Fragmentation ASDU Maximum Incoming Transfer Size. Please refer to [Z1] for further details on Fragmentation settings.

Any message that needs truncation shall truncate on a UTF-8 character boundary. The SE secure payload is 59 bytes for the Message field in a non- fragmented, non-source routed Display Message packet (11 bytes for other Display Message fields). Devices using fragmentation can send a message larger than this. Reserving bytes for source route will reduce this. InterPAN message payload for the “message” is 98 bytes.

10.5.2.3.2 Cancel Message Command

The Cancel Message command described in Figure 10-38 provides the ability to cancel the sending or acceptance of previously sent messages. When this message is received the recipient device has the option of clearing any display or user interfaces it supports, or has the option of logging the message for future reference.

Figure 10-38. Format of the *Cancel Message* Command Payload

Octets	4	1
Data Type	uint32	map8
Field Name	Message ID	Message Control

10.5.2.3.2.1 Payload Details

Message ID: A unique unsigned 32-bit number identifier for the message being cancelled. It's expected the value contained in this field is a unique number managed by upstream systems or a UTC based time stamp (UTC data type) identifying when the message was originally issued.

MessageControl: An enumerated field indicating the optional ability to pass the cancel message request onto the Anonymous Inter-PAN transmission mechanism. If the Anonymous Inter-PAN transmission mechanism is not supported on a particular device, this parameter is ignored. Bitmap values for this field are listed in Table 10-62.

10.5.3 Client

10.5.3.1 Dependencies

Support for ZCL Data Types.

No dependencies exist for other Smart Energy Clusters.

10.5.3.2 Attributes

None

10.5.3.3 Commands Generated

The command IDs generated by the Messaging cluster are listed in Table 10-63.

Table 10-63. Messaging Client Commands

Id	Description	M/O
0x00	*Get Last Message*	M
0x01	*Message Confirmation*	M

10.5.3.3.1 *GetLastMessage* Command

This command has no payload.

10.5.3.3.1.1 Effect on Receipt

On receipt of this command, the device shall send a Display Message command (refer to sub-clause 10.5.2.3.1). A ZCL Default Response with status NOT_FOUND shall be returned if no message is available.

10.5.3.3.2 MessageConfirmation Command

The Message Confirmation command described in Figure 10-39 provides the ability to acknowledge a previously sent message.

Figure 10-39. Format of the *Message Confirmation* Command Payload

Octets	4	4
Data Type	uint32	UTC
Field Name	Message ID	Confirmation Time

10.5.3.3.2.1 Payload Details

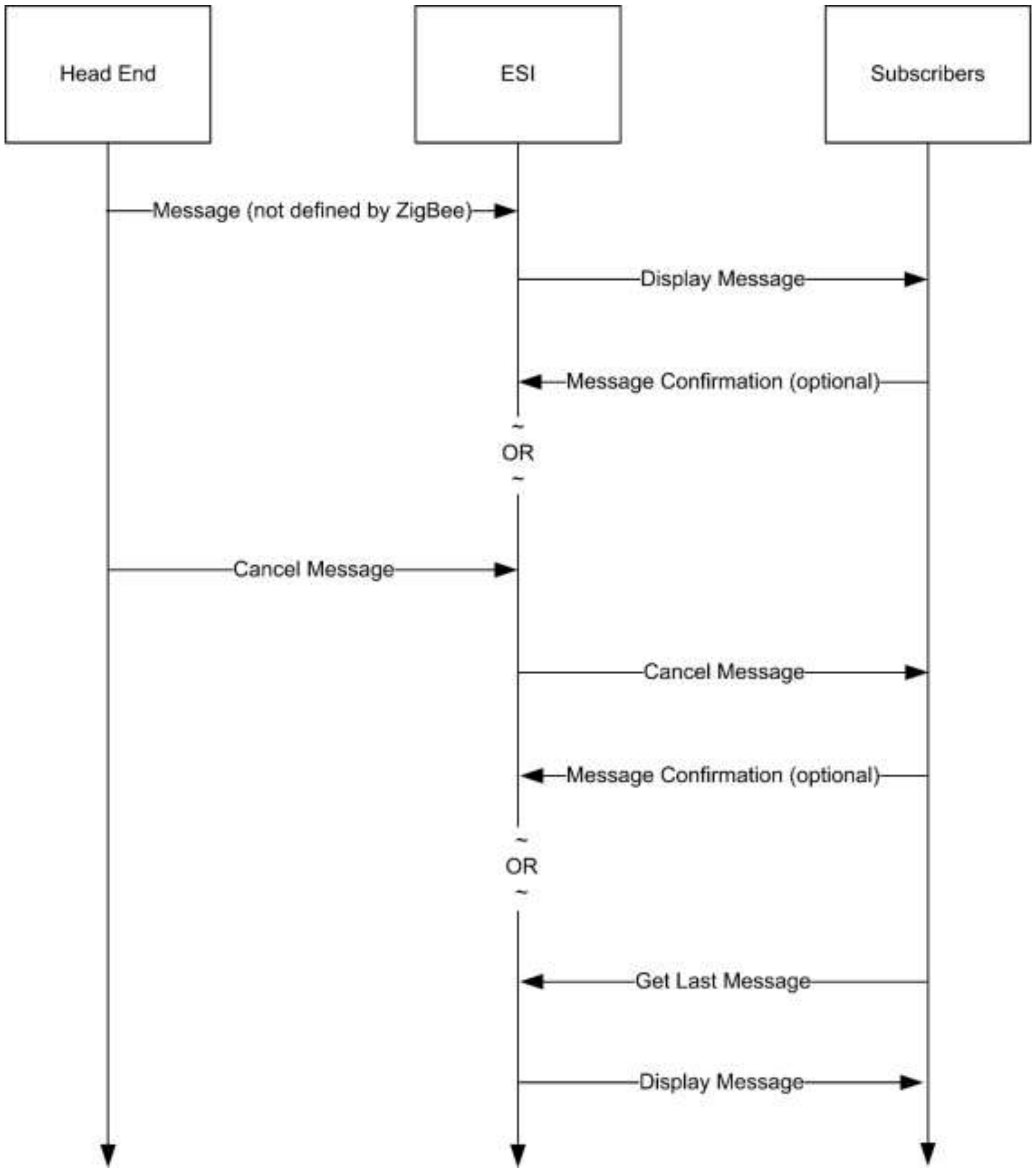
Message ID: A unique unsigned 32-bit number identifier for the message being confirmed.

Confirmation Time: UTC of user confirmation of message.

10.5.4 Application Guidelines

For Server and Client transactions, refer to Figure 10-40.

Figure 10-40. Client/Server Message Command Exchanges



10.6 Tunneling

Note: The optional support for flow control within the cluster in this revision of this specification is provisionary and not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

10.6.1 Overview

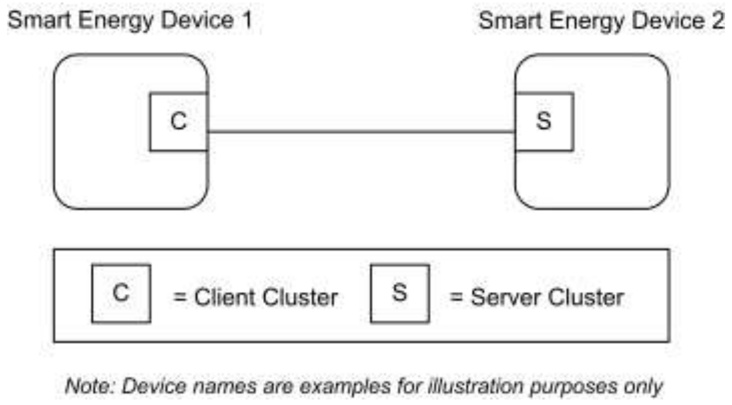
The tunneling cluster provides an interface for tunneling protocols. It is comprised of commands and attributes required to transport any existing metering communication protocol within the payload of standard ZigBee frames (including the handling of issues such as addressing, fragmentation and flow control). Examples for such protocols are DLMS/COSEM, IEC61107, ANSI C12, M-Bus, ClimateTalk, etc.

The tunneling foresees the roles of a server and a client taking part in the data exchange. Their roles are defined as follows:

Client: Requests a tunnel from the server and closes the tunnel if it is no longer needed.

Server: Provides and manages tunnels to the clients.

Figure 10-41. A Client Requests a Tunnel from a Server to Exchange Complex Data in Both Directions



The data exchange through the tunnel is symmetric. This means both client and server provide the commands to transfer data (TransferData). And both must make sure that only the partner to which the tunnel has been built up is granted RW access to it (e.g. tunnel identifier protection through checking the MAC address).

Sleepy devices either close the tunnel immediately after they have pushed their data through it, or leave it open in which case an attribute in the server (CloseTunnelTimeout) decides whether the tunnel is closed from the server side during the sleeping phase or not. If data is transferred to a non-existent or wrong tunnel identifier, the receiver generates an error message (TransferDataError).

The server may support more than one tunneling protocol. The type of tunnel to be opened is a mandatory parameter (ProtocolID) of the tunnel request (RequestTunnel) that the client needs to send to the server in order to set up a new tunnel. The response from the server (RequestTunnelResponse) will contain a parameter with the status of the tunnel (TunnelStatus). If the tunnel request was successful, a unique identifier (TunnelID) is returned within the response. In an error case (e.g. the requested protocol is not supported) the status contains the type of error. There is no special attribute in order to read out the supported protocols from the server. Either the client knows them a priori or it has to try several times using different ProtocolIDs until the server responds with the tunnel status Success.

The tunneling cluster adds optional support for flow control to handle streaming protocols such as IEC61107. If implemented, flow control messages are provided to control the data flow and send acknowledges to data messages on application level. However, flow control is an optional feature and disabled per default. In the default case, the acknowledge messages (AckTransferData) must not be sent in order to reduce complexity and prevent from unneeded overhead.

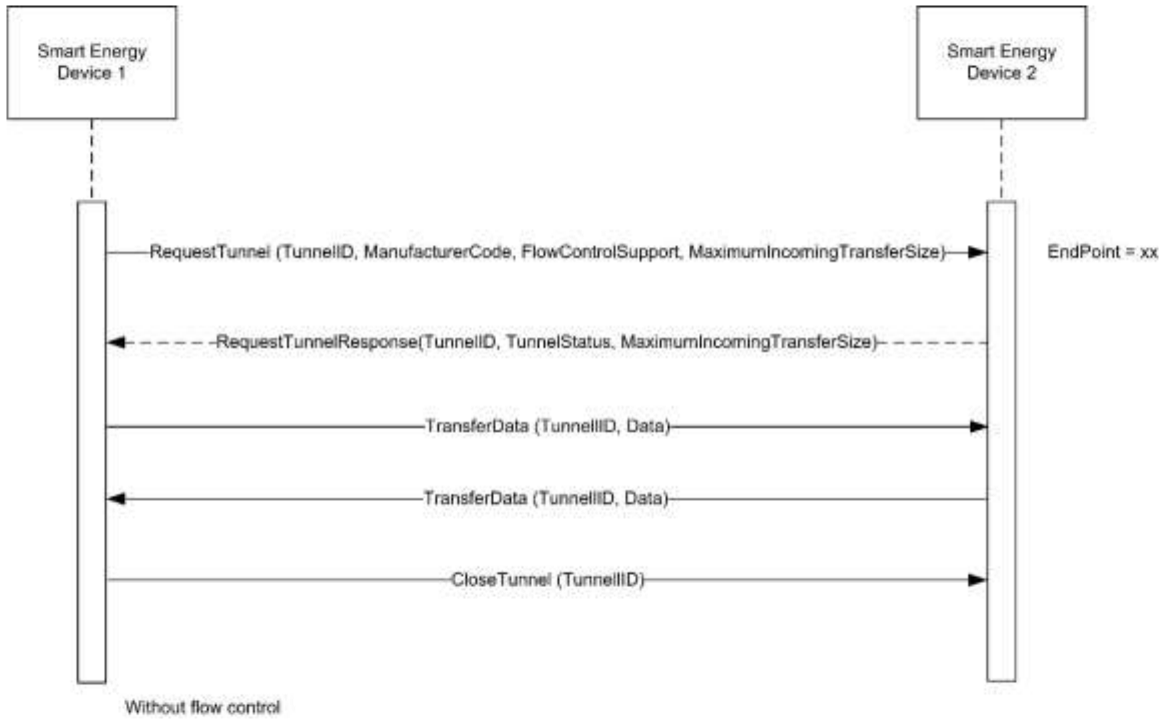
The following sequence describes a typical usage:

1. The client issues a service discovery to find devices which support the tunneling server cluster. The discovery may either be directed to one device, if its address is known, or be a broadcast (*MatchSimpleDescriptor*).

2. The response to the discovery from the server contains an endpoint number (*SimpleDescriptor*). Using this endpoint, the client directs a tunnel request to a given server. Together with the request, the client is required to provide an enumeration with the ID of the protocol that shall be tunneled. There is the possibility to request tunnels for manufacturer specific protocols. In this case, the *ProtocolID* has to be followed by a *ZigBee ManufacturerCode* to open the tunnel. An additional parameter for *FlowControlSupport* accompanies the request, together with an indication of the client's incoming buffer size (*RequestTunnel* (*ProtocolID*, *ManufacturerCode*, *FlowControlSupport*, *MaximumIncomingTransferSize*)).
3. If the server supports the protocol, it allocates the required resources, assigns a tunnel identifier and returns the ID number within the response including an additional tunnel status that the command was successful and the server's incoming buffer size. If the command failed, the status contains the reason in form of an error code (*RequestTunnelResponse* (*TunnelID*, *TunnelStatus*, *MaximumIncomingTransferSize*)). The tunnel identifier number would then be invalid in this case.
4. Both server and client may exchange data (*TransferData(Data)*). In case the optional flow control is utilized, each data transfer is acknowledged (*AckTransferData(NumberOfOctetsLeft)*). Additionally, there is the possibility to stop (*AckTransferData*(0)) and resume (*ReadyData(NumberOfOctetsLeft)* the data transfer.
5. After the transfer has been successfully completed, the client closes the tunnel again freeing the tunnel identifier in the server (*CloseTunnel(TunnelID)*). If not, the server closes the tunnel by itself after *CloseTunnelTimeout* seconds.

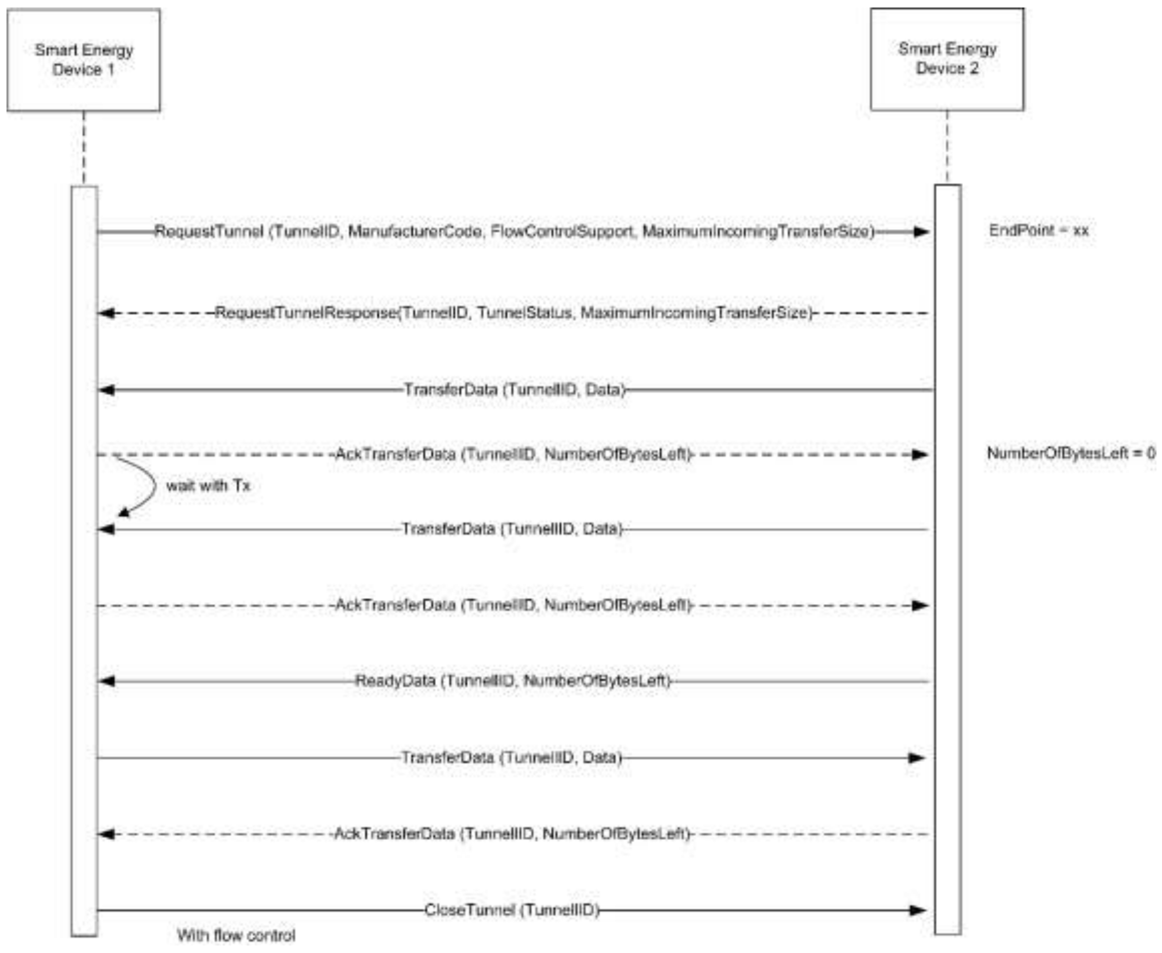
The following sequence diagrams show the client/server model and the typical usage of the cluster without flow control (Figure 10-42) and with flow control (Figure 10-43).

14837 **Figure 10-42. SE Device 1 (Client) Requests a Tunnel from SE Device 2 (Server) to Transfer Data Without Flow Control**
14838 **(Default)**


Smart Energy Device 1
Smart Energy Device 2
RequestTunnel (TunnelID, ManufacturerCode, FlowControlSupport, MaximumIncomingTransferSize)
EndPoint = xx
RequestTunnelResponse(TunnelID, TunnelStatus, MaximumIncomingTransferSize)
TransferData (TunnelID, Data)
TransferData (TunnelID, Data)
CloseTunnel (TunnelID)
Without flow control

14839

14840 **Figure 10-43. SE Device 1 (Client) Requests a Tunnel from SE Device 2 (Server) to Transfer Data with Flow Control**


Smart Energy Device 1
Smart Energy Device 2
RequestTunnel (TunnelID, ManufacturerCode, FlowControlSupport, MaximumIncomingTransferSize)
EndPoint = xx
RequestTunnelResponse(TunnelID, TunnelStatus, MaximumIncomingTransferSize)
TransferData (TunnelID, Data)
AckTransferData (TunnelID, NumberOfBytesLeft)
NumberOfBytesLeft = 0
wait with Tx
TransferData (TunnelID, Data)
AckTransferData (TunnelID, NumberOfBytesLeft)
ReadyData (TunnelID, NumberOfBytesLeft)
TransferData (TunnelID, Data)
AckTransferData (TunnelID, NumberOfBytesLeft)
CloseTunnel (TunnelID)
With flow control

14841

14842
10.6.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

14843
10.6.1.2 Classification

Hierarchy	Role	PICS Code
Base	Application	SETUN

14844
10.6.1.3 Cluster Identifiers

Identifier	Name
0x0704	Tunneling (Smart Energy)

10.6.2 Server

10.6.2.1 Dependencies

This cluster requires APS fragmentation [Z1] to be implemented, with maximum transfer sizes defined by the device's negotiated input buffer sizes.

10.6.2.2 Attributes

Table 10-64. Tunneling Cluster Attributes

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	*CloseTunnelTimeout*	uint16	0x0001-0xFFFF	R	0xFFFF	M

10.6.2.2.1 CloseTunnelTimeout Attribute

CloseTunnelTimeout defines the minimum number of seconds that the server waits on an inactive tunnel before closing it on its own and freeing its resources (without waiting for the CloseTunnel command from the client). Inactive means here that the timer is re-started with each new reception of a command. 0x0000 is an invalid value.

10.6.2.3 Parameters

Table 10-65 contains a summary of all parameters passed to or returned by the server commands. These values are considered as parameters (and not attributes) in order to facilitate the handling of the tunneling cluster for both the client and the server side. The parameters cannot be read or written via ZCL global commands. The detailed description of these parameters can be found in the according command sections of the document.

Table 10-65. Cluster Parameters Passed Through Commands

Name	Type	Range	Default	M/O
ProtocolID	enum8	0x01 – 0xFF	0x00	M
ManufacturerCode	uint16	0x0000 – 0xFFFF	0x00	M
FlowControlSupport	Boolean	TRUE or FALSE	FALSE	M
MaximumIncoming TransferSize	uint16	0x0000 – 0xFFFF	1500	M
TunnelID	uint16	0x0000 – 0xFFFF	(Return value)	M
Data	octstr	-	-	M
NumberOfOctetsLeft	uint16	0x0000 – 0xFFFF	-	M
TunnelStatus	uint8	0x00 – 0x04	-	M
TransferDataStatus	uint8	0x00 – 0x01	-	M

10.6.2.4 Commands Received

Table 10-66 lists cluster-specific commands received by the server.

Table 10-66. Cluster-specific Commands Received by the Server

Command Identifier FieldValue	Description	M/O
0x00	*RequestTunnel*	M
0x01	*CloseTunnel*	M
0x02	*TransferData*	M
0x03	*TransferDataError*	M
0x04	*AckTransferData*	O
0x05	*ReadyData*	O
0x06	*GetSupportedTunnelProtocols*	O

10.6.2.4.1 RequestTunnel Command

RequestTunnel is the client command used to setup a tunnel association with the server. The request payload specifies the protocol identifier for the requested tunnel, a manufacturer code in case of proprietary protocols and the use of flow control for streaming protocols.

10.6.2.4.1.1 Payload Format

Figure 10-44. Format of the *RequestTunnel* Command Payload

Octets	1	2	1	2
Data Type	enum8	uint16	bool	uint16
Field Name	ProtocolID (M)	Manufacturer Code (M)	FlowControl Support (M)	Maximum Incoming TransferSize

10.6.2.4.1.2 Payload Details

ProtocolID: An enumeration representing the identifier of the metering communication protocol for which the tunnel is requested. Table 10-67 lists the possible values for the ProtocolID. The values above 199 may be used for manufacturer-specific protocols.

Table 10-67. ProtocolID Enumerations

Values	Description
0	DLMS/COSEM (IEC 62056)
1	IEC 61107
2	ANSI C12
3	M-BUS
4	SML
5	ClimateTalk
200 to 254	Manufacturer-defined protocols

Manufacturer Code: A code that is allocated by the ZigBee Alliance, relating the manufacturer to a device and – for the tunneling - a manufacturer specific protocol. The parameter is ignored when the ProtocolID value is less than 200. This allows for 55 manufacturer-defined protocols for each manufacturer to be defined. A value of 0xFFFF indicates that the Manufacturer Code is not used.

FlowControlSupport: A Boolean type parameter that indicates whether flow control support is requested from the tunnel (TRUE) or not (FALSE). The default value is FALSE (no flow control).

MaximumIncomingTransferSize: A value that defines the size, in octets, of the maximum data packet that can be transferred to the client in the payload of a single TransferData command.

10.6.2.4.1.3 When Generated

Is never generated by the server.

10.6.2.4.1.4 Effect on Receipt

Triggers a process within the server to allocate resources and build up a new tunnel. A RequestTunnelResponse is generated and sent back to the client containing the result of the RequestTunnel command.

10.6.2.4.2 CloseTunnel Command

Client command used to close the tunnel with the server. The parameter in the payload specifies the tunnel identifier of the tunnel that has to be closed. The server leaves the tunnel open and the assigned resources allocated until the client sends the CloseTunnel command or the CloseTunnelTimeout fires.

10.6.2.4.2.1 Payload Format

Figure 10-45. Format of the *CloseTunnel* Command Payload

Octets	2
Data Type	uint16
Field Name	TunnelID (M)

10.6.2.4.2.2 Payload Details

TunnelID: The identifier of the tunnel that shall be closed. It is the same number that has been previously returned in the response to a RequestTunnel command. Valid numbers range between 0..65535 and must correspond to a tunnel that is still active and maintained by the server.

10.6.2.4.2.3 When Generated

This command is never generated by the server.

10.6.2.4.2.4 Effect on Receipt

In case the given TunnelID is correct, the server closes the tunnel and frees the resources. The associated tunnel is no longer maintained.

10.6.2.4.3 TransferData Command

Command that indicates (if received) that the client has sent data to the server. The data itself is contained within the payload.

10.6.2.4.3.1 Payload Format

Figure 10-46. Format of the *TransferData* Command Payload

Octets	2	Variable
Data Type	uint16	opaque
Field Name	TunnelID (M)	Data (M)

10.6.2.4.3.2 Payload Details

TunnelID: A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server triggered through the RequestTunnel command. This ID must be used to send data through the tunnel or passed with any commands concerning that specific tunnel.

Data: Series of octets containing the data to be transferred through the tunnel in the format of the communication protocol for which the tunnel has been requested and opened. The payload contains the assembled data exactly as it was sent by the client. Theoretically, its length is solely limited through the fragmentation algorithm and the RX/TX transfer buffer sizes within the communication partners. The content of the payload is up to the application sending the data. It is neither guaranteed, that it contains a complete PDU nor is any other assumption on its internal format made. This is left up to the implementer of the specific protocol tunnel behavior.

10.6.2.4.3.3 When Generated

Is generated whenever the server wants to tunnel protocol data to the client.

10.6.2.4.3.4 Effect on Receipt

Indicates that the server has received tunneled protocol data from the client.

10.6.2.4.4 TransferDataError Command

This command is generated by the receiver of a TransferData command if the tunnel status indicates that something is wrong. There are three cases in which TransferDataError is sent:

- The *TransferData* received contains a *TunnelID* that does not match to any of the active tunnels of the receiving device. This could happen if a (sleeping) device sends a *TransferData* command to a tunnel that has been closed by the server after the *CloseTunnelTimeout*.
- The *TransferData* received contains a proper *TunnelID* of an active tunnel, but the device sending the data does not match to it.
- The *TransferData* received contains more data than indicated by the *MaximumIncomingTransferSize* of the receiving device.

10.6.2.4.4.1 Payload Format

Figure 10-47. Format of the *TransferDataError* Command Payload

Octets	2	1
Data Type	uint16	uint8
Field Name	TunnelID (M)	TransferDataStatus (M)

10.6.2.4.4.2 Payload Details

TunnelID: A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server triggered through the RequestTunnel command. This ID must be used for the data transfer through the tunnel or passed with any commands concerning that specific tunnel.

TransferDataStatus: The TransferDataStatus parameter indicates the error that occurred within the receiver after the last TransferData command.

The TransferDataStatus values are shown in Table 10-68.

Table 10-68. TransferDataStatus Values

Value	Description	Remarks
0x00	No such tunnel	The *TransferData* command contains a *TunnelID* of a non-existent tunnel.
0x01	Wrong device	The *TransferData* command contains a *TunnelID* that does not match the device sending the data.
0x02	Data overflow	The *TransferData* command contains more data than indicated by the *MaximumIncomingTransferSize* of the receiving device

10.6.2.4.4.3 When Generated

Is generated if the server wants to tell the client that there was something wrong with the last TransferData command.

10.6.2.4.4.4 Effect on Receipt

Indicates that the client wants to tell the server that there was something wrong with the last TransferData command.

10.6.2.4.5 AckTransferData Command

Command sent in response to each TransferData command in case – and only in case – flow control has been requested by the client in the TunnelRequest command and is supported by both tunnel endpoints. The response payload indicates the number of octets that may still be received by the receiver.

10.6.2.4.5.1 Payload Format

Figure 10-48. Format of the *AckTransferData* Command Payload

Octets	2	2
Data Type	uint16	uint16
Field Name	TunnelID (M)	NumberOfBytesLeft (M)

10.6.2.4.5.2 Payload Details

TunnelID: A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server triggered through the RequestTunnel command. This ID must be used for the data transfer through the tunnel or passed with any commands concerning that specific tunnel.

NumberOfBytesLeft: Indicates the number of bytes that may still be received by the initiator of this command (receiver). It is most likely the remaining size of the buffer holding the data that is sent over TransferData. As an example: A value of 150 indicates that the next TransferData command must not contain more than 150 bytes of payload or data will get lost. A value of 0 indicates that there is no more space left in the receiver and the sender should completely stop sending data. After the reception of a ReadyData command, the sender may continue its data transfer.

10.6.2.4.5.3 When Generated

If flow control is on, the command is issued by the server to inform the client that the last TransferData command has been successfully received and how much space is left to receive further data.

10.6.2.4.5.4 Effect on Receipt

If flow control is on, the reception of this command indicates that the client wants to inform the server that the last TransferData command has been successfully received and how much space is left to receive further data.

10.6.2.4.6 ReadyData Command

The ReadyData command is generated – after a receiver had to stop the dataflow using the AckTransferData(0) command – to indicate that the device is now ready to continue receiving data. The parameter NumberOfOctetsLeft gives a hint on how much space is left for the next data transfer. The ReadyData command is only issued if flow control is enabled.

10.6.2.4.6.1 Payload Format

Figure 10-49. Format of the *ReadyData* Command Payload

Octets	2	2
Data Type	uint16	uint16
Field Name	TunnelID (M)	NumberOfOctetsLeft (M)

10.6.2.4.6.2 Payload Details

TunnelID: A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server triggered through the RequestTunnel command. This ID must be used for the data transfer through the tunnel or passed with any commands concerning that specific tunnel.

NumberOfOctetsLeft: Indicates the number of octets that may be received by the initiator of this command (receiver). It is most likely the remaining size of the buffer holding the data that is sent over TransferData. As an example: A value of 150 indicates that the next TransferData command must not contain more than 150 bytes of payload or data will get lost. The value must be larger than 0. As for its exact value, it is up to the implementer of the cluster to decide what flow control algorithm shall be applied.

10.6.2.4.6.3 When Generated

If generated by the server, this command informs the client that it may now continue to send and how much space is left within the server to receive further data.

10.6.2.4.6.4 Effect on Receipt

If received by the server, this command informs the server that it may now continue to send and how much space is left within the client to receive further data.

10.6.2.4.7 Get Supported Tunnel Protocols Command

Get Supported Tunnel Protocols is the client command used to determine the tunnel protocols supported on another device.

10.6.2.4.7.1 Payload Format

Figure 10-50. Format of the *Get Supported Tunnel Protocols* Command Payload

Octets	1
Data Type	uint8
Field Name	Protocol Offset

10.6.2.4.7.2 Payload Details

Protocol Offset: Where there are more protocols supported than can be returned in a single Supported Tunnel Protocols Response command, this field allows an offset to be specified on subsequent Get Supported Tunnel Protocols commands. An offset of zero (0x00) should be used for an initial (or only) Get Supported Tunnel Protocols command (indicating that the returned list of protocols should commence with first available protocol). As a further example, if 10 protocols had previously been returned, the next Get Supported Tunnel Protocols command should use an offset of 10 (0x0A) to indicate the 11th available protocol should be the first returned in the next response.

10.6.2.4.7.3 Effect on Receipt

On receipt of this command, a device will respond with a Supported Tunnel Protocols Response command, indicating the tunnel protocols it supports (see sub- clause 10.6.2.5.6 for further details).

10.6.2.5 Commands Generated

Table 10-69 lists commands that are generated by the server.

Table 10-69. Cluster-Specific Commands Sent by the Server

Command Identifier FieldValue	Description	M/O
0x00	*RequestTunnelResponse*	M
0x01	*TransferData*	M
0x02	*TransferDataError*	M
0x03	*AckTransferData*	O
0x04	*ReadyData*	O
0x05	*Supported Tunnel Protocols Response*	O
0x06	*TunnelClosureNotification*	O

10.6.2.5.1 RequestTunnelResponse Command

RequestTunnelResponse is sent by the server in response to a RequestTunnel command previously received from the client. The response contains the status of the RequestTunnel command and a tunnel identifier corresponding to the tunnel that has been set-up in the server in case of success.

10.6.2.5.1.1 Payload Format

Figure 10-51. Format of the *RequestTunnelResponse* Command Payload

Octets	2	1	2
Data Type	uint16	uint8	uint16
Field Name	TunnelID (M)	TunnelStatus (M)	Maximum Incoming TransferSize

10.6.2.5.1.2 Payload Details

TunnelID: A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server triggered through the RequestTunnel command. This ID must now be used to send data through this tunnel (TunnelID, TransferData) and is also required to close the tunnel again (CloseTunnel). If the command has failed, the TunnelStatus contains the reason of the error and the TunnelID is set to 0xFFFF.

TunnelStatus: The TunnelStatus parameter indicates the server's internal status after the execution of a RequestTunnel command.

The TunnelStatus values are shown in Table 10-70.

Table 10-70. *TunnelStatus* Values

Value	**Description**	**Remarks**
0x00	Success	The tunnel has been opened and may now be used to transfer data in both directions.
0x01	Busy	The server is busy and cannot create a new tunnel at the moment. The client may try again after a recommended timeout of 3 minutes.
0x02	No more tunnel IDs	The server has no more resources to setup requested tunnel. Clients should close any open tunnels before retrying.
0x03	Protocol not supported	The server does not support the protocol that has been requested in the ProtocolID parameter of the *RequestTunnel* command.
0x04	Flow control not supported	Flow control has been requested by the client in the *RequestTunnel* command but cannot be provided by the server (missing resources or no support).

MaximumIncomingTransferSize: A value that defines the size, in octets, of the maximum data packet that can be transferred to the server in the payload of a single TransferData command.

10.6.2.5.1.3 When Generated

Is generated in reply to a RequestTunnel command to inform the client about the result of the request.

10.6.2.5.1.4 Effect on Receipt

Should never be received by the server.

10.6.2.5.2 TransferData Command

Command that transfers data from server to the client. The data itself has to be placed within the payload.

10.6.2.5.2.1 Payload Format

Figure 10-52. Format of the *TransferData* Command Payload

Octets	2	Variable
Data Type	uint16	opaque
Field Name	TunnelID (M)	Data (M)

10.6.2.5.2.2 Payload Details

TunnelID: A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server triggered through the RequestTunnel command. This ID must be used for the data transfer through the tunnel or passed with any commands concerning that specific tunnel.

Data: Series of octets containing the data to be transferred through the tunnel in the format of the communication protocol for which the tunnel has been requested and opened. The payload containing the assembled data exactly as it has been sent away by the client. Theoretically, its length is solely limited through the fragmentation algorithm and the RX/TX transfer buffer sizes within the communication partners. The content of the payload is up to the application sending the data. It is not guaranteed that it contains a complete PDU, nor is any assumption to be made on its internal format (which is left up to the implementer of the specific tunnel protocol).

10.6.2.5.2.3 When Generated

Is generated when the server wants to tunnel protocol data to the client.

10.6.2.5.2.4 Effect on Receipt

Indicates that the server has received tunneled protocol data from the client.

10.6.2.5.3 TransferDataError Command

See sub-clause 10.6.2.4.4.

10.6.2.5.4 AckTransferData Command

See sub-clause 10.6.2.4.5.

10.6.2.5.5 ReadyData Command

See sub-clause 10.6.2.4.6.

10.6.2.5.6 Supported Tunnel Protocols Response Command

Supported Tunnel Protocols Response is sent in response to a Get Supported Tunnel Protocols command previously received. The response contains a list of tunnel protocols supported by the device; the payload of the response should be capable of holding up to 16 protocols.

10.6.2.5.6.1 Payload Format

Figure 10-53. Format of the *Supported Tunnel Protocols Response* Command Payload

Octets	1	1	3	…	3
Data Type	bool	uint8			
Field Name	Protocol List Complete	Protocol Count	Protocol 1	…	Protocol n

where each protocol field shall be formatted as:

Figure 10-54. Format of the Supported Tunnel Protocols Response Command Protocol Fields

Octets	2	1
Data Type	uint16	enum8
Field Name	Manufacturer Code	Protocol ID

10.6.2.5.6.2 Payload Details

Protocol List Complete: The Protocol List Complete field is a Boolean; a value of 0 indicates that there are more supported protocols available (if more than 16 protocols are supported). A value of 1 indicates that the list of supported protocols is complete.

Protocol Count: The number of Protocol fields contained in the response.

Manufacturer Code: A code that is allocated by the ZigBee Alliance, relating the manufacturer to a device and - for tunneling - a manufacturer specific protocol. A value of 0xFFFF indicates a standard (i.e. non- manufacturer specific) protocol

Protocol ID: An enumeration representing the identifier of the metering communication protocol for the supported tunnel. Table 10-67 lists the possible values for standard protocols

10.6.2.5.6.3 When Generated

Is generated in reply to a Get Supported Tunnel Protocols command, to indicate the tunnel protocols supported by the device

10.6.2.5.7 TunnelClosureNotification Command

TunnelClosureNotification is sent by the server to indicate that a tunnel has been closed due to expiration of a CloseTunnelTimeout.

10.6.2.5.7.1 Payload Format

Figure 10-55. Format of the *TunnelClosureNotification* Command Payload

Octets	2
Data Type	uint16
Field Name	TunnelID (M)

10.6.2.5.7.2 Payload Details

TunnelID: The identifier of the tunnel that has been closed. It is the same number that has been previously returned in the response to a RequestTunnel command. Valid numbers range between 0..65535 and must correspond to a tunnel that was still active and maintained by the server.

10.6.2.5.7.3 When Generated

The command is sent by a server when a tunnel is closed due to expiration of CloseTunnelTimeout. It is sent unicast to the client that had originally requested that tunnel.

10.6.3 Client

10.6.3.1 Dependencies

This cluster requires APS fragmentation [Z1] to be implemented, with maximum transfer sizes defined by the device's negotiated input buffer sizes.

10.6.3.2 Attributes

The client has no cluster specific attributes.

10.6.3.3 Commands Received

The client receives the cluster-specific response commands detailed in 10.6.2.5.

10.6.3.4 Commands Generated

The client generates the cluster-specific commands detailed in 10.6.2.4, as required by the application.

10.7 Key Establishment

10.7.1 Scope and Purpose

This section specifies a cluster that contains commands and attributes necessary for managing secure communication between devices.

This section should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see Chapter 2), which gives an overview of the library and specifies the frame formats and general commands used therein.

This version is specifically for inclusion in the Smart Energy profile. The document which originates from [Z10] will continue to be developed in a backward-compatible manner as a more general secure communication cluster for ZigBee applications as a whole.

10.7.2 General Description

10.7.2.1 Introduction

As previously stated, this document describes a cluster for managing secure communication in ZigBee. The cluster is for Key Establishment.

10.7.2.2 Network Security

The Key Establishment Cluster has been designed to be used where the underlying network security cannot be trusted. As such, no information that is confidential information will be transported.

10.7.2.3 Key Establishment

To allow integrity and confidentiality of data passed between devices, cryptographic schemes need to be deployed. The cryptographic scheme deployed in the ZigBee Specification for frame integrity and confidentiality is based upon a variant of the AES-CCM described in [N3] called AES-CCM*. This relies on the existence of secret keying material shared between the involved devices. There are methods to distribute this secret keying material in a trusted manner. However, these methods are generally not scalable or communication may be required with a trusted key allocation party over an insecure medium. This leads to the requirement for automated key establishment schemes to overcome these problems.

Key establishment schemes can be effected[123] using either a key agreement scheme or a key transport scheme. The key establishment scheme described in this document uses a key agreement scheme, therefore key transport schemes will not be considered further in this document.

A key agreement scheme is where both parties contribute to the shared secret and therefore the secret keying material to be established is not sent directly; rather, information is exchanged between both parties that allows each party to derive the secret keying material. Key agreement schemes may use either symmetric key or asymmetric key (public key) techniques. The party that begins a key agreement scheme is called the initiator, and the other party is called the responder.

Key establishment using key agreement involves an initiator and a responder and four steps:

1. Establishment of a trust relationship
2. Exchange of ephemeral data
3. Use of this ephemeral data to derive secret keying material using key agreement
4. Confirmation of the secret keying material.

There are two basic types of key establishment that can be implemented:

- Symmetric Key Key Establishment
- Public Key Key Establishment

10.7.2.4 Symmetric Key Key Establishment

Symmetric Key Key Establishment (SKKE) is based upon establishing a link key based on a shared secret (master key). If the knowledge of the shared secret is compromised, the established link key can also be compromised. If the master key is publicly known or is set to a default value, it is known as Unprotected Key Establishment (UKE). SKKE is the key establishment method used in the ZigBee specification therefore it will not be considered any further.

10.7.2.5 Public Key Key Establishment

Public Key Key Establishment (PKKE) is based upon establishing a link key based on shared static and ephemeral public keys. As the public keys do not require any secrecy, the established link key cannot be compromised by knowledge of them.

As a device's static public key is used as part of the link key creation, it can either be transported independently to the device's identity where binding between the two is assumed, or it can be transported as part of a implicit certificate signed by a Certificate Authority, which provides authentication of the binding between the device's identity and its public key as part of the key establishment process. This is called Certificate-Based Key Establishment (CBKE) and is discussed in more detail in sub-clause 10.7.4.2.

CBKE provides the most comprehensive form of Key Establishment and therefore will be the method specified in this cluster.

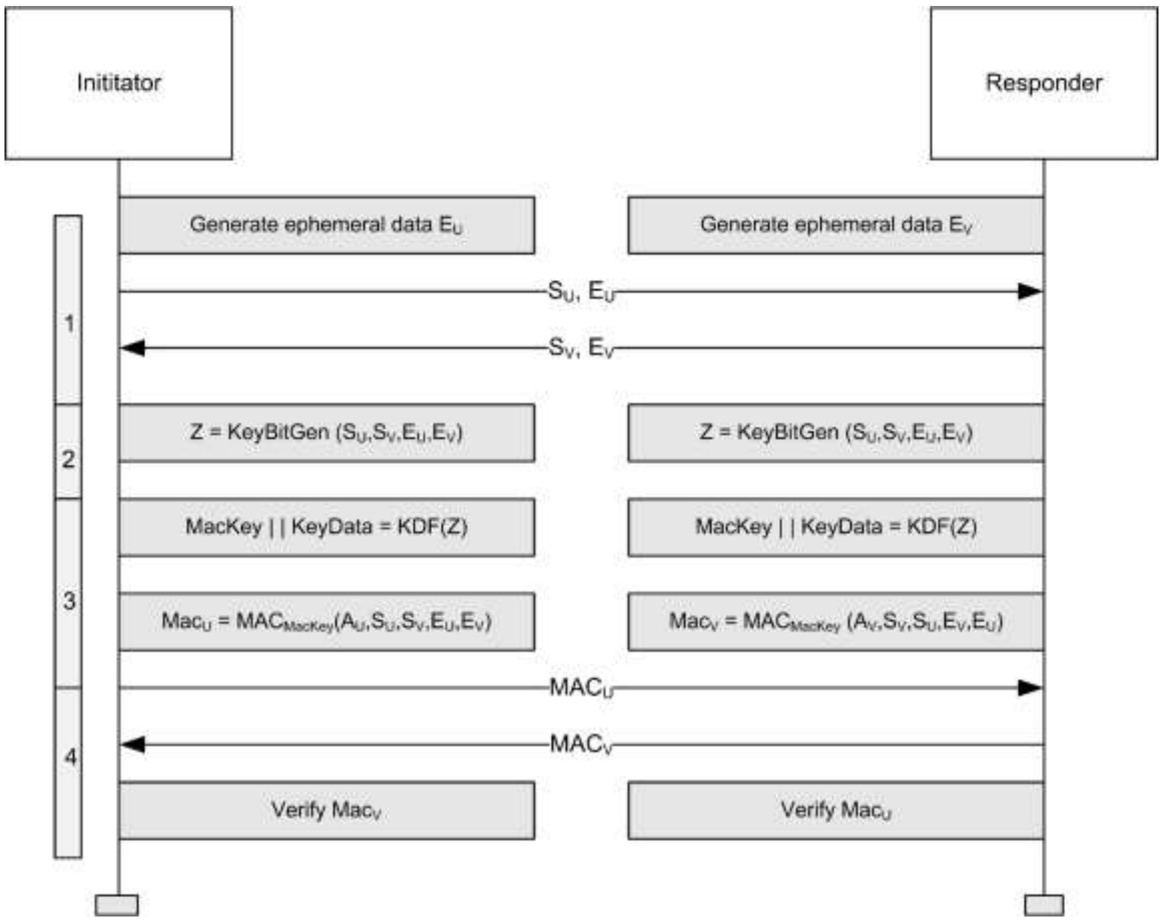
The purpose of the key agreement scheme as described in this document is to produce shared secret keying material which can be subsequently used by devices using AES-CCM* the cryptographic scheme deployed in the ZigBee Specification or for any proprietary security mechanism implemented by the application.

[123] CCB 2288

10.7.2.6 General Exchange

Figure 10-56 shows an overview of the general exchange which takes place between initiator and responder to perform key establishment.

Figure 10-56. Overview of General Exchange


Inititator
Responder
Generate ephemeral data E_U
Generate ephemeral data E_V
S_U, E_U
S_V, E_V
1
2
3
4
Z = KeyBitGen (S_U,S_V,E_U,E_V)
MacKey | | KeyData = KDF(Z)
Mac_U = MAC_{MacKey}(A_U,S_U,S_V,E_U,E_V)
Mac_V = MAC_{MacKey} (A_V,S_V,S_U,E_V,E_U)
MAC_U
MAC_V
Verify Mac_V
Verify Mac_U

The functions are:

1. Exchange Static and Ephemeral Data
2. Generate Key Bitstream
3. Derive MAC key and Key Data
4. Confirm Key using MAC

The functions shown in Figure 10-56 depend on the Key Establishment mechanism.

10.7.2.6.1 Exchange Static and Ephemeral Data

Figure 10-56 shows static data S_U and S_V. For PKKE schemes, this represents a combination of the 64-bit device address [Z11] and the device's static public key. The identities are needed by the MAC scheme and the static public keys are needed by the key agreement scheme.

Figure 10-56 also shows ephemeral data E_U and E_V. For PKKE schemes, this represents the public key of a randomly generated key pair.

The static and ephemeral data S_U and E_U are sent to V and the static and ephemeral data S_V and E_V and are sent to U.

10.7.2.6.2 Generate Key Bitstream

Figure 10-56 shows the KeyBitGen function for generating the key bitstream. The function's four parameters are the identifiers and the ephemeral data for both devices. This ensures the same key is generated at both ends.

For PKKE schemes, this is the ECMQV key agreement schemes specified in Section 6.2 of SEC1 [O1] . The static data S_U represents the static public key $Q_{1,U}$ of party U, the static data S_V represents the static public key $Q_{1,V}$ of party V, the ephemeral data E_U represents the ephemeral public key $Q_{2,U}$ of party U and the ephemeral data E_V represents the ephemeral public key $Q_{2,V}$ of party V.

10.7.2.6.3 Derive MAC Key and Key Data

Figure 10-56 shows the KDF (KeyDerivation Function) for generating the MAC Key and key data. The MAC Key is used with a keyed hash message authentication function to generate a MAC and the key data is the shared secret, e.g. the link key itself required for frame protection.

For PKKE schemes, this is the key derivation function as specified in Section 3.6.1 of SEC1 [O1]. Note there is no SharedInfo parameter of the referenced KDF, i.e. it is a null octet string of length 0.

Figure 10-56 also shows generation of the MAC using the MAC Key derived using the KDF using a message comprised of both static data S_U and S_V and ephemeral data E_U and E_V plus an additional component A which is different for initiator and responder.

For PKKE schemes, this is the MAC scheme specified in section 3.7 of SEC1 [O1]. The MAC in the reference is the keyed hash function for message authentication specified in sub-clause 10.7.4.2.2.6 and the message M is a concatenation of the identity (the 64-bit device address [E1]) of U, the identity of V and point-compressed octet-string representations of the ephemeral public keys of parties U and V. The order of concatenation depends on whether it is the initiator or responder. The additional component A is the single octet 02_{16} for the initiator and 03_{16} for the responder.

10.7.2.6.4 Confirm Key Using MAC

Figure 10-56 shows MACs MAC_U and MAC_V.

The MAC MAC_U is sent to V and the MAC MAC_V is sent to U. U and V both calculate the corresponding MAC and compare it with the data received.

10.7.3 Cluster List

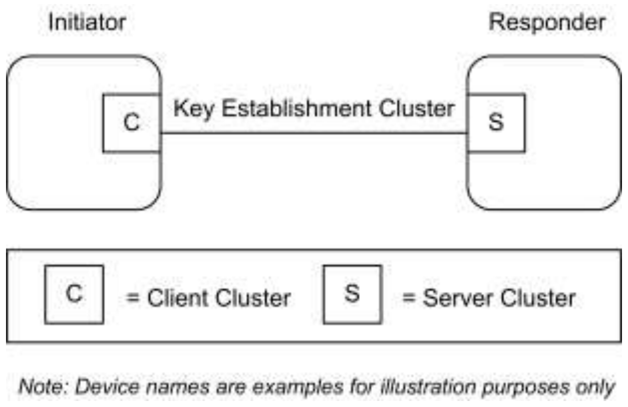
The clusters specified in this document are listed in Table 10-71.

For our purposes, any device that implements the client side of this cluster may be considered the initiator of the secure communication transaction.

Table 10-71. Clusters Specified for the Secure Communication Functional Domain

Cluster Name	Description
Key Establishment	Attributes and commands for establishing a shared secret between two ZigBee devices

Figure 10-57. Typical Usage of the Key Establishment Cluster


Initiator
Responder
C
Key Establishment Cluster
S
C = Client Cluster
S = Server Cluster
Note: Device names are examples for illustration purposes only

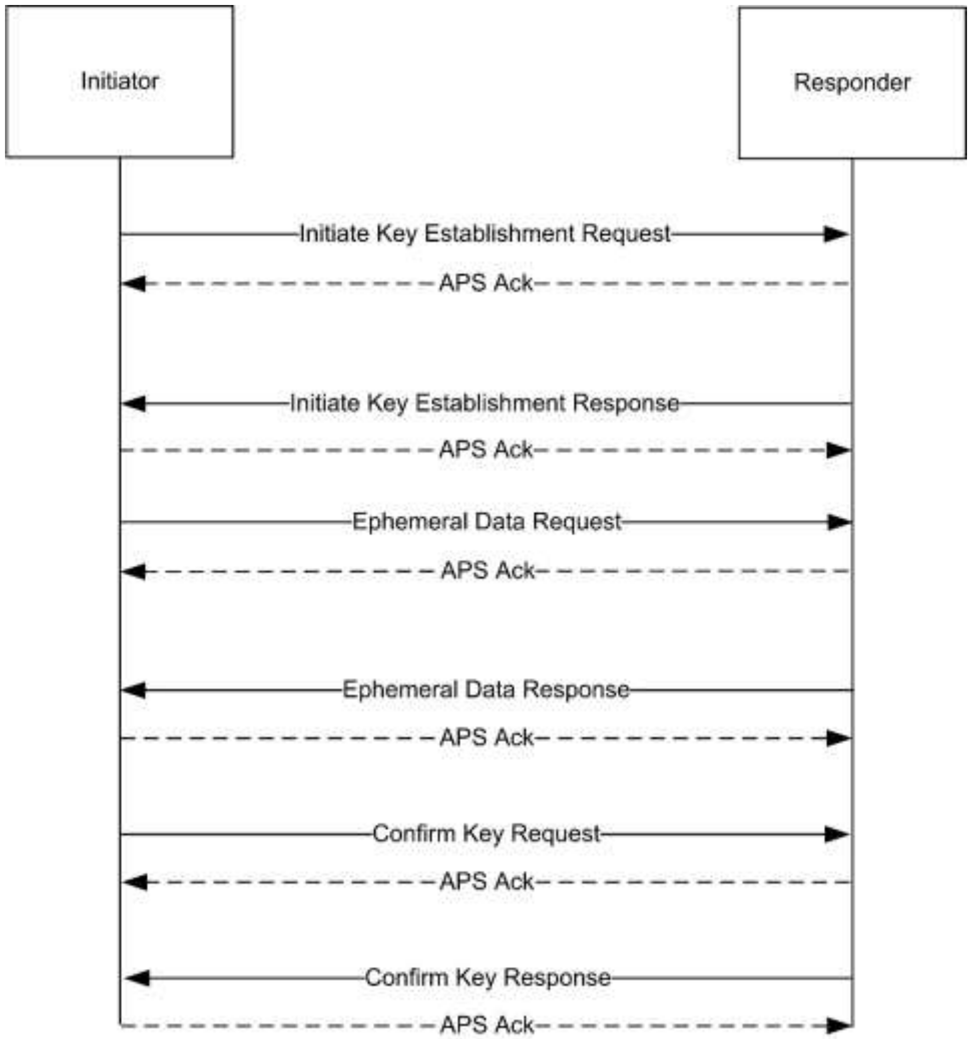
10.7.3.1 Key Establishment Cluster

10.7.3.1.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides attributes and commands to perform mutual authentication and establish keys between two ZigBee devices. Figure 10-58 depicts a diagram of a successful key establishment negotiation.

Figure 10-58. Key Establishment Command Exchange



As depicted above, all Key Establishment messages should be sent with APS retries enabled. A failure to receive an ACK in a timely manner can be seen as a failure of key establishment. No Terminate Key Establishment should be sent to the partner of device that has timed out the operation.

The initiator can initiate the key establishment with any active endpoint on the responder device that supports the key establishment cluster. The endpoint can be either preconfigured or discovered, for example, by using ZDO Match-Desc-req. A link key successfully established using key establishment is valid for all endpoints on a particular device. The responder shall respond to the initiator using the source endpoint of the initiator's messages as the destination endpoint of the responder's messages.

It is expected that the time it takes to perform the various cryptographic computations of the key establishment cluster may vary greatly based on the device. Therefore rather than set static timeouts, the Initiate Key Establishment Request and Response messages will contain approximate values for how long the device will take to generate the ephemeral data and how long the device will take to generate confirm key message.

A device performing key establishment can use this information in order to choose a reasonable timeout for its partner during those operations. The timeout should also take into consideration the time it takes for a message to traverse the network including APS retries. A minimum transmission time of 2 seconds is recommended.

For the Initiate Key Establishment Response message, it is recommended the initiator wait at least 2 seconds before timing out the operation. It is not expected that generating an Initiate Key Establishment Response will take significant time compared to generating the Ephemeral Data and Confirm Key messages.

10.7.3.1.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

10.7.3.1.1.2 Classification

Hierarchy	Role	PICS Code
Base	Application	SEKE

10.7.3.1.1.3 Cluster Identifiers

Identifier	Name
0x0800	Key Establishment (Smart Energy)

10.7.3.1.2 Server

10.7.3.1.2.1 Dependencies

The Key Establishment server cluster has no dependencies.

10.7.3.1.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 10-72.

Table 10-72. *Key Establishment* Attribute Sets

Attribute Set Identifier	Description
0x000	Information

10.7.3.1.2.2.1 Information

The *Information* attribute set contains the attributes summarized in Table 10-73.

Table 10-73. *Information* Attribute Sets

Id	Name	Type	Range	Access	Default	M/O
0x0000	*KeyEstablishmentSuite*	enum16	0x0000 - 0xFFFF	R	0x0000	M

10.7.3.1.2.2.1.1 KeyEstablishmentSuite Attribute

The KeyEstablishmentSuite attribute is 16-bits in length and specifies all the cryptographic schemes for key establishment on the device. A device shall set the corresponding bit to 1 for every cryptographic scheme that is supports. All other cryptographic schemes and reserved bits shall be set to 0.

Table 10-74. Values of the *KeyEstablishmentSuite* Attribute

Bits	Description
0	Certificate-based Key Establishment (CBKE-ECMQV)

10.7.3.1.2.3 Commands Received

The server side of the key establishment cluster is capable of receiving the commands listed in Table 10-75.

Table 10-75. Received Command IDs for the Key Establishment Cluster Server

Command Identifier Field	Description	M/O
0x00	*Initiate Key EstablishmentRequest*	M
0x01	*Ephemeral Data Request*	M
0x02	*Confirm Key Data Request*	M
0x03	*Terminate Key Establishment*	M

10.7.3.1.2.3.1 Initiate Key Establishment Request Command

The Initiate Key Establishment Request command allows a device to initiate key establishment with another device. The sender will transmit its identity information and key establishment protocol information to the receiving device.

10.7.3.1.2.3.1.1 Payload Format

The Initiate Key Establishment Request command payload shall be formatted as illustrated in Figure 10-59.

Figure 10-59. Format of the *Initiate Key Establishment Request* Command Payload

Octets	2	1	1	48
Data Type	map16	uint8	uint8	opaque
Field Name	Key Establishment suite	Ephemeral Data Generate Time	Confirm Key Generate Time	Identity (IDU)

Key Establishment Suite: This will be the type of Key Establishment that the initiator is requesting for the Key Establishment Cluster. For CBKE-ECMQV this will be 0x0001.

Ephemeral Data Generate Time: This value indicates approximately how long the initiator device will take in seconds to generate the Ephemeral Data Request command. The valid range is 0x00 to 0xFE.

Confirm Key Generate Time: This value indicates approximately how long the initiator device will take in seconds to generate the Confirm Key Request command. The valid range is 0x00 to 0xFE.

Identity field: For KeyEstablishmentSuite = 0x0001 (CBKE), the identity field shall be the block of octets containing the implicit certificate CERTU as specified in sub-clause 10.7.4.2.

10.7.3.1.2.3.1.2 Effect on Receipt

If the device does not currently have the resources to respond to a key establishment request it shall send a Terminate Key Establishment command with the result value set to NO_RESOURCES and the Wait Time field shall be set to an approximation of the time that must pass before the device will have the resources to process a new Key Establishment Request.

If the device can process this request, it shall check the Issuer field of the device's implicit certificate. If the Issuer field does not contain a value that corresponds to a known Certificate Authority, the device shall send a Terminate Key Establishment command with the result set to UNKNOWN_ISSUER.

If the device accepts the request it shall send an Initiate Key Establishment Response command containing its own identity information. The device should verify the certificate belongs to the address that the device is communicating with. The binding between the identity of the communicating device and its address is verifiable using out-of-band method.

10.7.3.1.2.3.2 Ephemeral Data Request Command

The Ephemeral Data Request command allows a device to communicate its ephemeral data to another device and request that the device send back its own ephemeral data.

10.7.3.1.2.3.2.1 Payload Format

Figure 10-60. Format of the *Ephemeral Data Request* Command Payload

Octets	22
Data Type	opaque
Field Name	Ephemeral Data (QEU)

10.7.3.1.2.3.2.2 Effect on Receipt

If the device is not currently in the middle of negotiating Key Establishment with the sending device when it receives this message, it shall send back a Terminate Key Establishment message with a result of BAD_MESSAGE. If the device is in the middle of Key Establishment with the sender but did not receive this message in response to an Initiate Key Establishment Response command, it shall send back a Terminate Key Establishment message with a result of BAD_MESSAGE. If the device can process the request it shall respond by generating its own ephemeral data and sending an Ephemeral Data Response command containing that value.

10.7.3.1.2.3.3 Confirm Key Request Command

The Confirm Key Request command allows the initiator sending device to confirm the key established with the responder receiving device based on performing a cryptographic hash using part of the generated keying material and the identities and ephemeral data of both parties.

10.7.3.1.2.3.3.1 Payload Format

The Confirm KeyRequest command payload shall be formatted as illustrated in Figure 10-61.

Figure 10-61. Format of the *Confirm Key Request* Command Payload

Octets	16
Data Type	opaque
Field Name	Secure Message Authentication Code (*MACU*)

Secure Message Authentication Code field: The Secure Message Authentication Code field shall be the octet representation of MACU as specified in sub-clause 10.7.4.2.

10.7.3.1.2.3.3.2 Effect on Receipt

If the device is not currently in the middle of negotiating Key Establishment with the sending device when it receives this message, it shall send back a Terminate Key Establishment message with a result of BAD_MESSAGE. If the device is in the middle of Key Establishment with the sender but did not receive this message in response to an Ephemeral Data Response command, it shall send back a Terminate Key Establishment message with a result of BAD_MESSAGE.

On receipt of the Confirm Key Request command the responder device shall compare the received MACU value with its own reconstructed version of MACU. If the two match the responder shall send back MACV by generating an appropriate Confirm Key Response command. If the two do not match, the responder shall send back a Terminate Key Establishment with a result of BAD KEY_CONFIRM and terminate the key establishment.

10.7.3.1.2.3.4 Terminate Key Establishment Command

The Terminate Key Establishment command may be sent by either the initiator or responder to indicate a failure in the key establishment exchange.

10.7.3.1.2.3.4.1 Payload Format

The Terminate Key Establishment command payload shall be formatted as illustrated in Figure 10-62.

Figure 10-62. Format of the *Terminate Key Establishment* Command Payload

Octets	1	1	2
Data Type	enum8	uint8	map16
Field Name	Status Code	Wait Time	KeyEstablishmentSuite

Status Field: The Status field shall be one of the error codes in Table 10-76.

Table 10-76. *Terminate Key Establishment* Command Status Field

Enumeration	**Value**	**Description**
UNKNOWN_ISSUER	0x01	The Issuer field within the key establishment partner's certificate is unknown to the sending device, and it has terminated the key establishment.
BAD_KEY_CONFIRM	0x02	The device could not confirm that it shares the same key with the corresponding device and has terminated the key establishment.
BAD_MESSAGE	0x03	The device received a bad message from the corresponding device (e.g. message with bad data, an out of sequence number, or a message with a bad format) and has terminated the key establishment.
NO_RESOURCES	0x04	The device does not currently have the internal resources necessary to perform key establishment and has terminated the exchange.
UNSUPPORTED_SUITE	0x05	The device does not support the specified key establishment suite in the partner's Initiate Key Establishment message.

Wait Time: This value indicates the minimum amount of time in seconds the initiator device should wait before trying to initiate key establishment again. The valid range is 0x00 to 0xFE.

KeyEstablishmentSuite: This value will be set the value of the KeyEstablishmentSuite attribute. It indicates the list of key exchange methods that the device supports.

10.7.3.1.2.3.4.2 Effect on Receipt

On receipt of the Terminate Key Establishment command the device shall terminate key establishment with the sender. If the device receives a status of BAD_MESSAGE or NO_RESOURCES it shall wait at least the time specified in the Wait Time field before trying to re-initiate Key Establishment with the device.

If the device receives a status of UNKNOWN_SUITE it should examine the KeyEstablishmentSuite field to determine if another suite can be used that is supported by the partner device. It may re-initiate key establishment using that one of the supported suites after waiting the amount of time specified in the Wait Time field. If the device does not support any of the types in the KeyEstablishmentSuite field, it should not attempt key establishment again with that device.

If the device receives a status of UNKNOWN_ISSUER or BAD_KEY_CONFIRM the device should not attempt key establishment again with the device, as it is unlikely that another attempt will be successful.

10.7.3.1.2.4 Commands Generated

The server generates the commands detailed in sub-clause 10.7.3.1.3.3, as well as those used for reading and writing attributes.

10.7.3.1.3 Client

10.7.3.1.3.1 Dependencies

The Key Establishment client cluster has no dependencies.

10.7.3.1.3.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 10-77.

Table 10-77. Key Establishment Attribute Sets

Attribute Set Identifier	Description
0x000	Information

10.7.3.1.3.2.1 Information

The Information attribute set contains the attributes summarized in Table 10-78.

Table 10-78. Attributes of the Information Attribute Set

Id	Name	Type	Range	Acc	Default	M/O
0x0000	*KeyEstablishmentSuite*	enum16	0x0000 – 0xFFFF	R	0x0000	M

10.7.3.1.3.2.1.1 KeyEstablishmentSuite Attribute

The KeyEstablishmentSuite attribute is 16-bits in length and specifies all the cryptographic schemes for key establishment on the device. A device shall set the corresponding bit to 1 for every cryptographic scheme that is supports. All other cryptographic schemes and reserved bits shall be set to 0. This attribute shall be set to one of the non-reserved values listed in Table 10-79.

Table 10-79. Values of the *KeyEstablishmentSuite* Attribute

KeyEstablishmentSuite	Description
0	Certificate-based Key Establishment (CBKE - ECMQV)

10.7.3.1.3.3 Commands Received

The client side of the Key Establishment cluster is capable of receiving the commands listed in Table 10-80.

Table 10-80. Received Command IDs for the Key Establishment Cluster Client

Command Identifier Field Value	Description	M/O
0x00	*Initiate Key Establishment Response*	M
0x01	*Ephemeral Data Response*	M
0x02	*Confirm Key Data Response*	M
0x03	*Terminate Key Establishment*	M

10.7.3.1.3.3.1 Initiate Key Establishment Response Command

The Initiate Key Establishment Response command allows a device to respond to a device requesting the initiation of key establishment with it. The sender will transmit its identity information and key establishment protocol information to the receiving device.

10.7.3.1.3.3.1.1 Payload Format

The Initiate Key Establishment Response command payload shall be formatted as illustrated in Figure 10-63.

Figure 10-63. Format of the *Initiate Key Establishment Response* Command Payload

Octets	2	1	1	48
Data Type	map16	uint8	uint8	opaque
Field Name	Requested Key Establishment suite	Ephemeral Data Generate Time	Confirm Key Generate Time	Identity (IDU)

Requested Key Establishment Suite: This will be the type of KeyEstablishmentSuite that the initiator has requested be used for the key establishment exchange. The device shall set a single bit in the bitmask indicating the requested suite, all other bits shall be set to zero.

Ephemeral Data Generate Time: This value indicates approximately how long in seconds the responder device takes to generate the Ephemeral Data Response message. The valid range is 0x00 to 0xFE.

Confirm Key Generate Time: This value indicates approximately how long the responder device will take in seconds to generate the Confirm Key Response message. The valid range is 0x00 to 0xFE.

Identity field: For KeyEstablishmentSuite = 0x0001 (CBKE), the identity field shall be the block of Octets containing the implicit certificate CERTU as specified in sub-clause 10.7.4.2.

10.7.3.1.3.3.1.2 Effect on Receipt

If the device is not currently in the middle of negotiating Key Establishment with the sending device when it receives this message, it shall send back a Terminate Key Establishment message with a result of BAD_MESSAGE. If the device is in the middle of Key Establishment with the sender but did not receive this message in response to an Initiate Key Establishment Request command, it shall send back a Terminate Key Establishment message with a result of BAD_MESSAGE.

On receipt of this command the device shall check the Issuer field of the device's implicit certificate. If the Issuer field does not contain a value that corresponds to a known Certificate Authority, the device shall send a Terminate Key Establishment command with the status value set to UNKNOWN_ISSUER. If the device does not currently have the resources to respond to a key establishment request it shall send a Terminate Key Establishment command with the status value set to NO_RESOURCES and the Wait Time field shall be set to an approximation of the time that must pass before the device has the resources to process the request.

If the device accepts the response it shall send an Ephemeral Data Request command. The device should verify the certificate belongs to the address that the device is communicating with. The binding between the identity of the communicating device and its address is verifiable using out-of-band method.

10.7.3.1.3.3.2 Ephemeral Data Response Command

The Ephemeral Data Response command allows a device to communicate its ephemeral data to another device that previously requested it.

10.7.3.1.3.3.2.1 Payload Format

Figure 10-64. Format of the *Ephemeral Data Response* Command Payload

Octets	22
Data Type	opaque
Field Name	Ephemeral Data (QEV)

10.7.3.1.3.3.2.2 Effect on Receipt

If the device is not currently in the middle of negotiating Key Establishment with the sending device when it receives this message, it shall send back a Terminate Key Establishment message with a result of BAD_MESSAGE. If the device is in the middle of Key Establishment with the sender but did not receive this message in response to an Ephemeral Data Request command, it shall send back a Terminate Key Establishment message with a result of BAD_MESSAGE.

On receipt of this command if the device can handle the request it shall perform key generation, key derivation, and MAC generation. If successful it shall generate an appropriate Confirm Key Request command, otherwise it shall generate a Terminate Key Establishment with a result value of NO_RESOURCES.

10.7.3.1.3.3.3 Confirm Key Response Command

The Confirm Key Response command allows the responder to verify the initiator has derived the same secret key. This is done by sending the initiator a cryptographic hash generated using the keying material and the identities and ephemeral data of both parties.

10.7.3.1.3.3.3.1 Payload Format

The Confirm Key Response command payload shall be formatted as illustrated in Figure 10-65.

Figure 10-65. Format of the *Confirm Key Response* Command Payload

Octets	16
Data Type	opaque
Field Name	Secure Message Authentication Code (*MACV*)

Secure Message Authentication Code field: The Secure Message Authentication Code field shall be the octet representation of MACV as specified in sub-clause 10.7.4.2.

10.7.3.1.3.3.3.2 Effect on Receipt

If the device is not currently in the middle of negotiating Key Establishment with the sending device when it receives this message, it shall send back a Terminate Key Establishment message with a result of BAD_MESSAGE. If the device is in the middle of Key Establishment with the sender but did not receive this message in response to a Confirm Key Request command, it shall send back a Terminate Key Establishment message with a result of BAD_MESSAGE.

On receipt of the Confirm Key Response command the initiator device shall compare the received MACV value with its own reconstructed version of the MACV. If the two match then the initiator can consider the key establishment process to be successful. If the two do not match, the initiator should send a Terminate Key Establishment command with a result of BAD_KEY_CONFIRM.

10.7.3.1.3.3.4 Terminate Key Establishment Command

The Terminate Key Establishment command may be sent by either the initiator or responder to indicate a failure in the key establishment exchange.

10.7.3.1.3.3.4.1 Payload Format

Figure 10-66. Format of the *Terminate Key Establishment* Command Payload

Octets	1	1	2
Data Type	enum8	uint8	map16
Field Name	Status Code	Wait Time	KeyEstablishmentSuite

Status field: The Status field shall be one of the error codes shown in Table 10-81.

Table 10-81. *Terminate Key Establishment* Command Status Field

Enumeration	Value	Description
UNKNOWN_ISSUER	0x01	The Issuer field within the key establishment partner's certificate is unknown to the sending device, and it has terminated the key establishment.
BAD_KEY_CONFIRM	0x02	The device could not confirm that it shares the same key with the corresponding device and has terminated the key establishment.
BAD_MESSAGE	0x03	The device received a bad message from the corresponding device (e.g. message with bad data, an out of sequence number, or a message with a bad format) and has terminated the key establishment.
NO_RESOURCES	0x04	The device does not currently have the internal resources necessary to perform key establishment and has terminated the exchange.
UNSUPPORTED_SUITE	0x05	The device does not support the specified key establishment suite in the partner's Initiate Key Establishment message.

Wait Time: This value indicates the minimum amount of time in seconds the initiator device should wait before trying to initiate key establishment again. The valid range is 0x00 to 0xFE.

KeyEstablishmentSuite: This value will be set the value of the KeyEstablishmentSuite attribute. It indicates the list of key exchange methods that the device supports.

10.7.3.1.3.3.4.2 Effect on Receipt

On receipt of the Terminate Key Establishment command the device shall terminate key establishment with the sender. If the device receives a status of BAD_MESSAGE or NO_RESOURCES it shall wait at least the time specified in the Wait Time field before trying to re-initiate Key Establishment with the device.

If the device receives a status of UNKNOWN_SUITE it should examine the KeyEstablishmentSuite field to determine if another suite can be used that is supported by the partner device. It may re-initiate key establishment using that one of the supported suites after waiting the amount of time specified in the Wait Time field. If the device does not support any of the types in the KeyEstablishmentSuite field, it should not attempt key establishment again with that device.

If the device receives a status of UNKNOWN_ISSUER or BAD_KEY_CONFIRM the device should not attempt key establishment again with the device, as it is unlikely that another attempt will be successful.

10.7.3.1.3.4 Commands Generated

The client generates the commands detailed in sub-clause 10.7.3.1.2.3, as well as those used for reading and writing attributes.

10.7.4 Application Implementation

10.7.4.1 Network Security for Smart Energy Networks

The underlying network security for Smart Energy networks is assumed to be ZigBee Standard security using pre-configured link keys.

A temporary link key for a joining device is produced by performing the cryptographic hash function on a random number assigned to the joining device (e.g. serial number) and the device identifier, which is the device's 64-bit IEEE address [Z11].

The joining device's assigned random number is then conveyed to the utility via an out-of-band mechanism (e.g. telephone call, or web site registration). The utility then commissions the energy service interface (ESI) at the premises where the joining device is by installing the temporary link key at the ESI on the back channel.

When the joining device powers up, it will also create a temporary link key as above and therefore at the time of joining both the joining device and the ESI have the same temporary link key, which can be used to transport the network key securely to the joining device.

At this point, the device will be considered joined and authenticated as far as network security is concerned. The secure communication cluster can now be invoked to replace the temporary link key with a more secure link key based on public key cryptography.

10.7.4.2 Certificate-Based Key Establishment

The Certificate-Based Key-Establishment (CBKE) solution uses public-key technology with digital certificates and root keys. Each device has a private key and a digital certificate that is signed by a Certificate Authority (CA).

The digital certificate includes:

- Reconstruction data for the device's public key
- The device's extended 64-bit IEEE address
- Profile specific information (e.g., the device class, network id, object type, validity date, etc.)

Certificates provide a mechanism for cryptographically binding a public key to a device's identity and characteristics.

Trust for a CBKE solution is established by provisioning a CA root key and a digital certificate to each device. A CA root key is the public key paired with the CA's private key. A CA uses its private key to sign digital certificates and the CA root key is used to verify these signatures. The trustworthiness of a public key is confirmed by verifying the CA's signature of the digital certificate. Certificates can be issued either by the device manufacturer, the device distributor, or the end customer. For example, in practical situations, the CA may be a computer (with appropriate key management software) that is kept physically secure at the end customer's facility or by a third-party.

At the end of successful completion of the CBKE protocol the following security services are offered:

- Both devices share a secret link key.
- Implicit Key Authentication: Both devices know with whom they share this link key.
- Key Confirmation: Each device knows that the other device actually has computed the key correctly.
- No Unilateral Key Control: No device has complete control over the shared link key that is established.
- Perfect Forward Secrecy: If the private key gets compromised none of future and past communications are exposed.
- Known Key Security resilience: Each shared link key created per session is unique.

10.7.4.2.1 Notation and Representation

10.7.4.2.1.1 Strings and String Operations

A string is a sequence of symbols over a specific set (e.g., the binary alphabet {0,1} or the set of all octets). The length of a string is the number of symbols it contains (over the same alphabet). The right-concatenation of two strings x and y of length m and n respectively (notation: x || y), is the string z of length m+n that coincides with x on its leftmost m symbols and with y on its rightmost n symbols. An octet is a bit string of length 8.

10.7.4.2.1.2 Integers and Their Representation

Throughout this specification, the representation of integers as bit strings or octet strings shall be fixed. All integers shall be represented as binary strings in most- significant-bit first order and as octet strings in most-significant-octet first order. This representation conforms to the convention in Section 2.3 of SEC1 [O1].

10.7.4.2.1.3 Entities

Throughout this specification, each entity shall be a DEV and shall be uniquely identified by its 64-bit IEEE device address [Z11]. The parameter entlen shall have the integer value 64.

10.7.4.2.2 Cryptographic Building Blocks

The following cryptographic primitives and data elements are defined for use with the CBKE protocol specified in this document.

10.7.4.2.2.1 Elliptic-Curve Domain Parameters

The elliptic curve domain parameters used in this specification shall be those for the curve “ansit163k1” as specified in section 3.4.1 of SEC2 [O2].

All elliptic-curve points (and operations here on) used in this specification shall be (performed) on this curve.

10.7.4.2.2.2 Elliptic-Curve Point Representation

All elliptic-curve points shall be represented as point compressed octet strings as specified in sections 2.3.3 and 2.3.4 of SEC1 [O1]. Thus, each elliptic-curve point can be represented in 22 bytes.

10.7.4.2.2.3 Elliptic-Curve Key Pair

An elliptic-curve-key pair consists of an integer d and a point Q on the curve determined by multiplying the generating point G of the curve by this integer (i.e., Q=dG) as specified in section 3.2.1 of SEC1 [O1]. Here, Q is called the public key, whereas d is called the private key; the pair (d, Q) is called the key pair. Each private key shall be represented as specified in section 2.3.7 of SEC1 [O1]. Each public key shall be represented as defined in sub-clause 10.7.4.2.1.2.

10.7.4.2.2.4 ECC Implicit Certificates

The exact format of the 48-byte implicit certificate IC_U used with CBKE scheme shall be specified as follows:

IC_U = PublicReconstrKey || Subject || Issuer || ProfileAttributeData

Where,

1. *PublicReconstrKey*: the 22-byte representation of the public-key reconstruction data BEU as specified in the implicit certificate generation protocol, which is an elliptic-curve point as specified in sub-clause 10.7.4.2.2.2 (see SEC4 [O1]);
2. *Subject*: the 8-byte identifier of the entity *U* that is bound to the public-key reconstruction data *BEU* during execution of the implicit certificate generation protocol (i.e., the extended, 64-bit IEEE 802.15.4 address [E1] of the device that purportedly owns the private key corresponding to the public key that can be reconstructed with *PublicReconstrKey*);
3. *Issuer***:** the 8-byte identifier of the CA that creates the implicit certificate during the execution of the implicit certificate generation protocol (the so-called Certificate Authority).
4. *ProfileAttributeData***:** the 10-byte sequence of octets that can be used by a ZigBee profile for any purpose. The first two bytes of this sequence is reserved as a profile identifier, which must be defined by another ZigBee standard.
5. The string I_U as specified in Step 6 of the actions of the CA in the implicit certificate generation protocol (see section SEC4 [O3]) shall be the concatenation of the *Subject*, *Issuer*, and *ProfileAttributeData*:

   I_U = *Subject* || *Issuer* || *ProfileAttributeData*

10.7.4.2.2.5 Block-Cipher

The block-cipher used in this specification shall be the Advanced Encryption Standard AES-128, as specified in FIPS Pub 197 [N4]. This block-cipher has a key size that is equal to the block size, in bits, i.e., keylen= 128.

10.7.4.2.2.6 Cryptographic Hash Function

The cryptographic hash function used in this specification shall be the blockcipher based cryptographic hash function specified in Annex B.6 in [Z1], with the following instantiations:

6. Each entity shall use the block-cipher E as specified in sub-clause B.1.1 in [Z1].
7. All integers and octets shall be represented as specified in sub-clause 10.7.4.2.1.

The Matyas-Meyer-Oseas hash function (specified in Annex B.6 in [Z1]) has a message digest size hashlen that is equal to the block size, in bits, of the established blockcipher.

10.7.4.2.2.7 Keyed Hash Function for Message Authentication

The keyed hash message authentication code (HMAC) used in this specification shall be HMAC, as specified in the FIPS Pub 198 [N5] with the following instantiations:

1. Each entity shall use the cryptographic hash *H* function as specified in sub-clause 10.7.4.2.2.6;
2. The block size *B* shall have the integer value 16 (this block size specifies the length of the data integrity key, in bytes, that is used by the keyed hash function, i.e., it uses a 128-bit data integrity key). This is also *MacKeyLen*, the length of *MacKey*.
3. The output size *HMAClen* of the HMAC function shall have the same integer value as the message digest parameter *hashlen* as specified in sub-clause 10.7.4.2.2.6.

10.7.4.2.2.8 Derived Shared Secret

The derived shared secret KeyData is the output of the key establishment. KeyData shall have length KeyDataLen of 128 bits.

10.7.4.2.3 Certificate-Based Key-Establishment

The CBKE method is used when the authenticity of both parties involved has not been established and where implicit authentication of both parties is required prior to key agreement.

The CBKE protocol has an identical structure to the PKKE protocol, except that implicit certificates are used rather than manual certificates. The implicit certificate protocol used with CBKE shall be the implicit certificate scheme with associated implicit certificate generation scheme and implicit certificate processing transformation as specified in SEC4 [O1], with the following instantiations:

1. Each entity shall be a DEV;
2. Each entity's identifier shall be its 64-bit device address [Z11]]; the parameter *entlen* shall have the integer value 64;
3. Each entity shall use the cryptographic hash function as specified in sub-clause 10.7.4.2.2.6;

The following additional information shall have been unambiguously established between devices operating the implicit certificate scheme:

1. Each entity shall have obtained information regarding the infrastructure that will be used for the operation of the implicit certificate scheme - including a certificate format and certificate generation and processing rules (see SEC4 [O1]);
2. Each entity shall have access to an authentic copy of the elliptic-curve public keys of one or more certificate authorities that act as CA for the implicit certificate scheme (SEC4 [O1]).

The methods by which this information is to be established are outside the scope of this standard.

The methods used during the CBKE protocol are described below. The parameters used by these methods are described in Table 10-82.

Table 10-82. Parameters Used by Methods of the CBKE Protocol

Parameter	Size (Octets)	Description
CERTU	48	The initiator device's implicit certificate used to transfer the initiator device's public key (denoted $Q_{1,U}$ in the Elliptic Curve MQV scheme in SEC1 [O1]) and the initiator device's identity.
CERTV	48	The responder device's implicit certificate used to transfer the responder device's public key (denoted $Q_{1,V}$ in the Elliptic Curve MQV scheme in SEC1 [O1]) and the responder device's identity.
QEU	22	The ephemeral public key generated by the initiator device (denoted $Q_{2,U}$ in the Elliptic Curve MQV scheme in SEC1 [O1]).
QEV	22	The ephemeral public key generated by the responder device (denoted Q_{2V} in the Elliptic Curve MQV scheme in SEC1 [O1]).
MACU	16	The secure message authentication code generated by the initiator device (where the message M is $(02_{16} \parallel ID_U \parallel ID_V \parallel QEU \parallel QEV)$ and ID_U and ID_V are the initiator and responder device entities respectively as specified in sub-clause C.4.2.2.3 and *QEU* and *QEV* are the point-compressed elliptic curve points representing the ephemeral public keys of the initiator and responder respectively as specified in sub-clause 10.7.4.2.2.2. See also section 3.7 of SEC1 [O1]).
MACV	16	The secure message authentication code generated by the responder device (where the message M is $(03_{16} \parallel ID_V \parallel ID_U \parallel QEV \parallel QEU)$ and ID_V and ID_U are the responder and initiator device entities respectively as specified in sub-clause C.4.2.2.3 and *QEV* and *QEU* are the point-compressed elliptic curve points representing the ephemeral public keys of the responder and initiator respectively as specified in sub-clause 10.7.4.2.2.3. See also section 3.7 of SEC1 [O1]).

10.7.4.2.3.1 Exchange Ephemeral Data

10.7.4.2.3.1.1 Initiator

The initiator device's implicit certificate CERTU and a newly generated ephemeral public key QEU are transferred to the responder device using the Initiate Key Establishment command via the Key Establishment Cluster Client.

10.7.4.2.3.1.2 Responder

The responder device's implicit certificate CERTV and a newly generated ephemeral public key QEV are transferred to the initiator device using the Initiate Key Establishment response command via the Key Establishment Cluster Server.

10.7.4.2.3.2 Validate Implicit Certificates

10.7.4.2.3.2.1 Initiator

The initiator device's Key Establishment Cluster Client processes the Initiate Key Establishment response command. The initiator device examines CERTV (formatted as IC_V as described in sub-clause 10.7.4.2.2.4), confirms that the Subject identifier is the purported owner of the certificate, and runs the certificate processing steps described in section SEC4 [O2].

10.7.4.2.3.2.2 Responder

The responder device's Key Establishment Cluster Server processes the Initiate Key Establishment command. The responder device examines CERTU (formatted as IC_U as described in sub-clause 10.7.4.2.2.4), confirms that the Subject identifier is the purported owner of the certificate, and runs the certificate processing steps described in section SEC 4 [O2].

10.7.4.2.3.3 Derive Keying Material

10.7.4.2.3.3.1 Initiator

The initiator performs the Elliptic Curve MQV scheme as specified in section 6.2 of SEC1 [O1] with the following instantiations:

1. The elliptic curve domain parameters shall be as specified in sub-clause 10.7.4.2.2.1;
2. The KDF shall use the cryptographic hash function specified in sub-clause 10.7.4.2.2.2;
3. The static public key $Q_{1,U}$ shall be the static public key of the initiator;
4. The ephemeral public key $Q_{2,U}$ shall be an ephemeral public key of the initiator generated as part of this transaction;
5. The static public key $Q_{1,V}$ shall be the static public key of the responder obtained from the responder's certificate communicated to the initiator by the responder;
6. The ephemeral public key $Q_{2,V}$ shall be based on the point-compressed octet string representation *QEV* of an ephemeral key of the responder communicated to the initiator by the responder;
7. The KDF parameter *keydatalen* shall be *MacKeyLen* + *KeyDataLen*, where *MacKeyLen* is the length of *MacKey* and *KeyDataLen* is the length of *KeyData*;
8. The parameter *SharedInfo* shall be the empty string.

The initiator device derives the keying material MacKey and KeyData from the output K as specified in section 3.6.1 of SEC1 [O1] by using MacKey as the leftmost MacKeyLen octets of K and KeyData as the rightmost KeyDataLen octets of K. KeyData is used subsequently as the shared secret and MacKey is used for key confirmation.

10.7.4.2.3.3.2 Responder

The responder performs the Elliptic Curve MQV scheme as specified in section 6.2 of SEC1 [O1] with the following instantiations:

1. The elliptic curve domain parameters shall be as specified in sub-clause 10.7.4.2.2.1;
2. The KDF shall use the cryptographic hash function specified in sub-clause 10.7.4.2.2.2;
3. The static public key $Q_{1,U}$ shall be the static public key of the initiator obtained from the initiator's certificate communicated to the responder by the initiator;
4. The ephemeral public key $Q_{2,U}$ shall be based on the point-compressed octet string representation *QEU* of an ephemeral key of the initiator communicated to the responder by the initiator;
5. The static public key $Q_{1,V}$ shall be the static public key of the responder;
6. The ephemeral public key $Q_{2,V}$ shall be an ephemeral public key of the responder generated as part of this transaction;
7. The KDF parameter *keydatalen* shall be *MacKeyLen* + *KeyDataLen*, where *MacKeyLen* is the length of *MacKey* and *KeyDataLen* is the length of *KeyData*;
8. The parameter *SharedInfo* shall be the empty string.

The responder device derives the keying material MacKey and KeyData from the output K as specified in section 3.6.1 of SEC1 [O1] by using MacKey as the leftmost MacKeyLen octets of K and KeyData as the rightmost KeyDataLen octets of K. KeyData is used subsequently as the shared secret and MacKey is used for key confirmation.

10.7.4.2.3.4 Confirm Keys

10.7.4.2.3.4.1 Initiator

The initiator device uses MacKey to compute its message authentication code MACU and sends it to the responder device by using the Confirm Key command via the Key Establishment Cluster Client.

The initiator device uses MacKey to confirm the authenticity of the responder by calculating MACV and comparing it with that sent by the responder.

10.7.4.2.3.4.2 Responder

The responder device uses MacKey to compute its message authentication code MACV and sends it to the initiator device by using the Confirm Key response command via the Key Establishment Cluster Server.

The responder device uses MacKey to confirm the authenticity of the initiator by calculating MACU and comparing it with that sent by the initiator.

10.7.5 Key Establishment Test Vectors

The following details the key establishment exchange data transformation and validation of test vectors for a pair of Smart Energy devices using Certificate based key exchange (CBKE) using Elliptical Curve Cryptography (ECC).

10.7.5.1 Preconfigured Data

Each device is expected to have been preinstalled with security information prior to initiating key establishment. The preinstalled data consists of the Certificate Authority's Public Key, a device specific certificate, and a device specific private key.

10.7.5.1.1 CA Public Key

The following is the Certificate Authority's Public Key.

```
02 00 FD E8 A7 F3 D1 08

42 24 96 2A 4E 7C 54 E6

9A C3 F0 4D A6 B8
```

10.7.5.1.2 Responder Data

The following is the certificate for device 1. The device has an IEEE of (>) 0000000000000001, and will be the responder.

```
03 04 5F DF C8 D8 5F FB

8B 39 93 CB 72 DD CA A5

5F 00 B3 E8 7D 6D 00 00

00 00 00 00 00 01 54 45
```

```
53 54 53 45 43 41 01 09

00 06 00 00 00 00 00 00
```

The certificate has the following data embedded within it:

Public Key Reconstruction Data	03 04 5F DF C8 D8 5F FB 8B 39 93 CB 72 DD CA A5 5F 00 B3 E8 7D 6D
Subject (IEEE)	00 00 00 00 00 00 00 01
Issuer	54 45 53 54 53 45 43 41
Attributes	01 09 00 06 00 00 00 00 00 00

The private key for device 1 is as follows:

```
00 b8 a9 00 fc ad eb ab

bf a3 83 b5 40 fc e9 ed

43 83 95 ea a7
```

The public key for device 1 is as follows:

```
03 02 90 a1 f5 c0 8d ad

5f 29 45 e3 35 62 0c 7a

98 fa c4 66 66 a1
```

10.7.5.1.3 Initiator Data

The following is the certificate for device 2. The device has an IEEE of (>) 0000000000000002, and will be the initiator.

```
02 06 15 E0 7D 30 EC A2

DA D5 80 02 E6 67 D9 4B

C1 B4 22 39 83 07 00 00

00 00 00 00 00 02 54 45

53 54 53 45 43 41 01 09

00 06 00 00 00 00 00 00
```

The certificate has the following data embedded within it:

Public Key Reconstruction Data	02 06 15 E0 7D 30 EC A2 DA D5 80 02 E6 67 D9

Subject (IEEE)	00 00 00 00 00 00 00 02
Issuer	54 45 53 54 53 45 43 41
Attributes	01 09 00 06 00 00 00 00 00 00

The private key for device 2 is as follows:

```
01 E9 DD B5 58 0C F7 2E

CE 7F 21 5F 0A E5 94 E4

8D F3 E7 FE E8
```

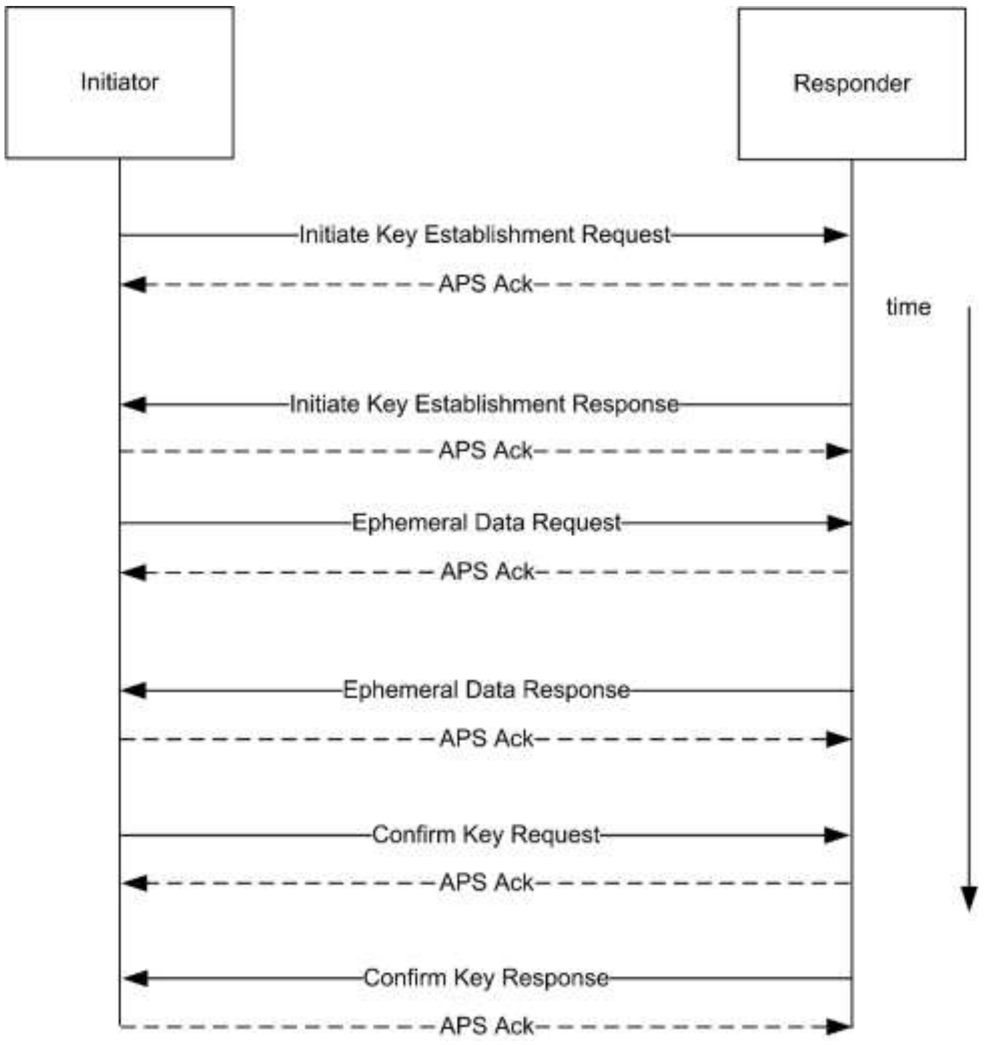
The public key for device 2 is:

```
03 02 5B BA 38 D0 C7 B5

43 6B 68 DF 72 8F 09 3E

7A 1D 6C 43 7E 6D
```

10.7.5.2 Key Establishment Messages

Figure 10-67 shows the basic flow of messages back and forth between the initiator and the responder performing key establishment using the Key Establishment Cluster.

Figure 10-67. Key Establishment Command Exchange



10.7.5.2.1 Initiate Key Establishment Request

The following is the APS message sent by the initiator (device 2) to the responder (device 1) for the initiate key establishment request.

```
40 0A 00 08 09 01 0A 01

01 00 00 01 00 03 06 02

06 15 E0 7D 30 EC A2 DA

D5 80 02 E6 67 D9 4B C1

B4 22 39 83 07 00 00 00

00 00 00 00 02 54 45 53

54 53 45 43 41 01 09 00
```

```
06 00 00 00 00 00 00
```

APS Header

Frame Control	0x40
Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x01

ZCL Header

Frame Control	0x01	Client to Server
Sequence Number	0x00	
Command Identifier	0x00	*Initiate Key Establishment Request*
Key Establishment Suite	0x0001	ECMQV
Ephemeral Data Generate Time	0x03	
Confirm Key Generate Time	0x06	
Identity (IDU)	*	Device 2’s certificate

10.7.5.2.2 Initiate Key Establishment Response

The following is the APS message sent by the responder (device 1) to the initiator (device 2) for the initiate key establishment response.

```
40 0A 00 08 09 01 0A 01

09 00 00 01 00 03 06 03

04 5F DF C8 D8 5F FB 8B

39 93 CB 72 DD CA A5 5F

00 B3 E8 7D 6D 00 00 00

00 00 00 00 01 54 45 53

54 53 45 43 41 01 09 00

06 00 00 00 00 00 00
```

APS Header

Frame Control	0x40

Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x01

ZCL Header

Frame Control	0x09	Server to Client
Sequence Number	0x00	
Command Identifier	0x00	*Initiate Key Establishment Response*
Key Establishment Suite	0x0001	ECMQV
Ephemeral Data Generate Time	0x03	
Confirm Key Generate Time	0x06	
Identity (IDV)	*	Device 1's certificate

10.7.5.2.3 Ephemeral Data Request

The following is the APS message sent by the initiator to the responder for the ephemeral data request.

```
40 0A 00 08 09 01 0A 02
01 01 01 03 00 E1 17 C8
6D 0E 7C D1 28 B2 F3 4E
90 76 CF F2 4A F4 6D 72
88
```

APS Header

Frame Control	0x40
Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x02

ZCL Header

Frame Control	0x01	Client to Server
Sequence Number	0x01	

Command Identifier	0x01	*Ephemeral Data Request*
Ephemeral Data (QEU)	`03 00 E1 17 C8 6D 0E 7C D1 28 B2 F3 4E 90 76 CF F2 4A F4 6D 72 88`	

10.7.5.2.4 Ephemeral Data Response

The following is the APS message sent by the responder to the initiator for the ephemeral data response.

```
40 0A 00 08 09 01 0A 02
09 01 01 03 06 AB 52 06
22 01 D9 95 B8 B8 59 1F
3F 08 6A 3A 2E 21 4D 84
5E
```

APS Header

Frame Control	0x40
Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x02

ZCL Header

Frame Control	0x09	Server to Client
Sequence Number	0x01	
Command Identifier	0x01	*Ephemeral Data Response*
Ephemeral Data (QEV)	`03 06 AB 52 06 22 01 D9 95 B8 B8 59 1F 3F 08 6A 3A 2E 21 4D 84 5E`	

10.7.5.2.5 Confirm Key Request

The following is the APS message sent by the initiator to the responder for the confirm key request.

```
40 0A 00 08 09 01 0A 03
01 02 02 B8 2F 1F 97 74
74 0C 32 F8 0F CF C3 92
1B 64 20
```

APS Header

Frame Control	0x40
Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x02

ZCL Header

Frame Control	0x01	Client to Server
Sequence Number	0x02	
Command Identifier	0x02	*Confirm Key Request*
Secure Message Authentication Code (MACU)	`B8 2F 1F 97 74 74 0C 32` `F8 0F CF C3 92 1B 64 20`	

10.7.5.2.6 Confirm Key Response

The following is the APS message sent by the responder to the initiator for the confirm key response.

```
40 0A 00 08 09 01 0A 03

09 02 02 79 D5 F2 AD 1C

31 D4 D1 EE 7C B7 19 AC

68 3C 3C
```

APS Header

Frame Control	0x40
Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x02

ZCL Header

Frame Control	0x09	Server to Client
Sequence Number	0x02	

Command Identifier	0x02	*Confirm Key Response*
Secure Message Authentication Code (MACV)	`79 D5 F2 AD 1C 31 D4 D1 EE 7C B7 19 AC 68 3C 3C`	

10.7.5.3 Data Transformation

The following are the various values used by the subsequent transformation.

U	Initiator
V	Responder
M(U)	Initiator Message Text (0x02)
M(V)	Responder Message Text (0x03)
ID(U)	Initiator's Identifier (IEEE address)
ID(V)	Responder's Identifier (IEEE address)
E(U)	Initiator's Ephemeral Public Key
E(V)	Responder's Ephemeral Public Key
E-P(U)	Initiator's Ephemeral Private Key
E-P(V)	Responder's Ephemeral Private Key
CA	Certificate Authority's Public Key
Cert(U)	Initiator's Certificate
Cert(V)	Responder's Certificate
Private(U)	Initiator's Private Key
Private(V)	Responder's Private Key
Shared Data	A pre-shared secret. NULL in Key Establishment
Z	A shared secret

Note: '||' stands for bitwise concatenation

10.7.5.3.1 ECMQV Primitives

It is assumed that an ECC library is available for creating the shared secret given the local private key, local ephemeral public & private key, remote device's certificate, remote device's ephemeral public key, and the certificate authority's public key. Further it is assumed that this library has been separately validated with a set of ECC test vectors. Those test vectors are outside the scope of this document.

10.7.5.3.2 Key Derivation Function (KDF)

Once a shared secret (Z) is established, a transform is done to create a SMAC (Secure Message Authentication Code) and a shared ZigBee Key.

10.7.5.3.3 Initiator Transform

Upon receipt of the responder's ephemeral data response, the initiator has all the data necessary to calculate the shared secret and derive the data for the confirm key request (SMAC).

10.7.5.3.3.1 Ephemeral Data

Public Key	`03 00 E1 17 C8 6D 0E 7C D1 28 B2 F3 4E 90 76 CF F2 4A F4 6D 72 88`
Private Key	`00 13 D3 6D E4 B1 EA 8E 22 73 9C 38 13 70 82 3F 40 4B FF 88 62`

10.7.5.3.3.2 Step Summary

1. Derive the Shared Secret using the ECMQV primitives

   Z = ECC_GenerateSharedSecret(Private(U), E(U), E-P(U), Cert(V), E(V), CA)
2. Derive the Keying data

   Hash-1 = Z || 00 00 00 01 || SharedData

   Hash-2 = Z || 00 00 00 02 || SharedData
3. Parse KeyingData as follows

   MacKey = First 128 bits (Hash-1) of KeyingData

   KeyData = Second 128 bits (Hash-2) of KeyingData
4. Create MAC(U)

   MAC(U) = MAC(MacKey) { M(U) || ID(U) || ID(V) || E(U) || E(V) }
5. Send MAC(U) to V.
6. Receive MAC(V) from V.
7. Calculate MAC(V)'

   MAC(V) = MAC(MacKey) { M(V) || ID(V) || ID(U) || E(V) || E(U) }
8. Verify MAC(V)' is the same as MAC(V).

10.7.5.3.3.3 Detailed Steps

1. Derive the Shared Secret using the ECMQV primitives

   Z = ECC_GenerateSharedSecret(Private(U), E(U), E-P(U), Cert(V), E(V), CA)

   00 E0 D2 C3 CC D5 C1 06 A8 9C 4F 6C C2 6A 5F 7E

   C9 DF 78 A7 BE
2. Derive the Keying data

   Hash-1 = Z || 00 00 00 01 || SharedData

Concatenation

```
00 E0 D2 C3 CC D5 C1 06 A8 9C 4F 6C C2 6A 5F 7E

C9 DF 78 A7 BE 00 00 00 01
```

Hash

```
90 F9 67 B2 2C 83 57 C1 0C 1C 04 78 8D E9 E8 48
```

Hash-2 = Z || 00 00 00 02 || SharedData

Concatenation

```
00 E0 D2 C3 CC D5 C1 06 A8 9C 4F 6C C2 6A 5F 7E
C9 DF 78 A7 BE 00 00 00 02
```

Hash

```
86 D5 8A AA 99 8E 2F AE FA F9 FE F4 96 06 54 3A
```

3. Parse KeyingData as follows

   MacKey = First 128 bits (Hash-1) of KeyingData

   KeyData = Second 128 bits (Hash-2) of KeyingData

4. Create MAC(U)

   MAC(U) = MAC(MacKey) { M(U) || ID(U) || ID(V) || E(U) || E(V) }

Concatenation

```
02 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00
01 03 00 E1 17 C8 6D 0E 7C D1 28 B2 F3 4E 90 76
CF F2 4A F4 6D 72 88 03 06 AB 52 06 22 01 D9 95
B8 B8 59 1F 3F 08 6A 3A 2E 21 4D 84 5E 88 00 10
```

Hash

```
B8 2F 1F 97 74 74 0C 32 F8 0F CF C3 92 1B 64 20
```

5. Send MAC(U) to V.
6. Receive MAC(V) from V.
7. Calculate MAC(V)'

   MAC(V) = MAC(MacKey) { M(V) || ID(V) || ID(U) || E(V) || E(U) }

Concatenation

```
03 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00
02 03 06 AB 52 06 22 01 D9 95 B8 B8 59 1F 3F 08
6A 3A 2E 21 4D 84 5E 03 00 E1 17 C8 6D 0E 7C D1
28 B2 F3 4E 90 76 CF F2 4A F4 6D 72 88 88 00 10
```

Hash

```
79 D5 F2 AD 1C 31 D4 D1 EE 7C B7 19 AC 68 3C 3C
```

8. Verify MAC(V)' is the same as MAC(V).

10.7.5.3.4 Responder Transform

Upon receipt of the initiator's confirm key request, the responder has all the data necessary to calculate the shared secret, validate the initiator's confirm key message, and derive the data for the confirm key response (SMAC).

10.7.5.3.4.1 Ephemeral Data

Public Key	03 06 AB 52 06 22 01 D9 95 B8 B8 59 1F 3F 08 6A 3A 2E 21 4D 84 5E
Private Key	03 D4 8C 72 10 DD BC C4 FB 2E 5E 7A 0A A1 6A 0D B8 95 40 82 0B

10.7.5.3.4.2 Step Summary

1. Derive the Shared Secret using the ECMQV primitives

   Z = ECC_GenerateSharedSecret(Private(V), E(V), E-P(V), Cert(U), E(U), CA)

2. Derive the Keying data

   Hash-1 = Z || 00 00 00 01 || SharedData

   Hash-2 = Z || 00 00 00 02 || SharedData

3. Parse KeyingData as follows

   MacKey = First 128 bits (Hash-1) of KeyingData

   KeyData = Second 128 bits (Hash-2) of KeyingData

4. Create MAC(V)

   MAC(V) = MAC(MacKey) { M(V) || ID(V) || ID(U) || E(V) || E(U) }

5. Calculate MAC(U)'

   MAC(U) = MAC(MacKey) { M(U) || ID(U) || ID(V) || E(U) || E(V) }

6. Verify MAC(U) is the same as MAC(U).

7. Send MAC(V) to U.

10.7.5.3.4.3 Detailed Steps

8. Derive the Shared Secret using the ECMQV primitives

   Z = ECC_GenerateSharedSecret(Private(U), E(U), E-P(U), Cert(V), E(V), CA)

```
00 E0 D2 C3 CC D5 C1 06 A8 9C 4F 6C C2 6A 5F 7E
C9 DF 78 A7 BE
```

9. Derive the Keying data

   Hash-1 = Z || 00 00 00 01 || SharedData

Concatenation

```
00 E0 D2 C3 CC D5 C1 06 A8 9C 4F 6C C2 6A 5F 7E

C9 DF 78 A7 BE 00 00 00 01
```

Hash

```
90 F9 67 B2 2C 83 57 C1 0C 1C 04 78 8D E9 E8 48
```

Hash-2 = Z || 00 00 00 02 || SharedData

Concatenation

```
00 E0 D2 C3 CC D5 C1 06 A8 9C 4F 6C C2 6A 5F 7E
C9 DF 78 A7 BE 00 00 00 02
```

Hash

```
86 D5 8A AA 99 8E 2F AE FA F9 FE F4 96 06 54 3A
```

10. Parse KeyingData as follows

    MacKey = First 128 bits (Hash-1) of KeyingData

    KeyData = Second 128 bits (Hash-2) of KeyingData

11. Create MAC(V)

    MAC(V) = MAC(MacKey) { M(V) || ID(V) || ID(U) || E(V) || E(U) }

Concatenation

```
03 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00
02 03 06 AB 52 06 22 01 D9 95 B8 B8 59 1F 3F 08
6A 3A 2E 21 4D 84 5E 03 00 E1 17 C8 6D 0E 7C D1
28 B2 F3 4E 90 76 CF F2 4A F4 6D 72 88 88 00 10
```

Hash

```
79 D5 F2 AD 1C 31 D4 D1 EE 7C B7 19 AC 68 3C 3C
```

12. Calculate MAC(V)'

    MAC(U) = MAC(MacKey) { M(U) || ID(U) || ID(V) || E(U) || E(V) }

Concatenation

```
02 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00
01 03 00 E1 17 C8 6D 0E 7C D1 28 B2 F3 4E 90 76
CF F2 4A F4 6D 72 88 03 06 AB 52 06 22 01 D9 95
B8 B8 59 1F 3F 08 6A 3A 2E 21 4D 84 5E 88 00 10
```

Hash

```
B8 2F 1F 97 74 74 0C 32 F8 0F CF C3 92 1B 64 20
```

13. Verify MAC(V) is the same as MAC(V).
14. Send MAC(V) to U.

CHAPTER 11 OVER-THE-AIR UPGRADE

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

11.1 Introduction

11.1.1 Purpose

The objective of this chapter is to provide detailed technical requirements for Over-The-Air image upgrade. This chapter presents a clear methodology for implementation of the OTA Upgrade cluster using the existing ZigBee stack(s), ZigBee Cluster Library and this OTA cluster.

The main goal of Over-The-Air Upgrade cluster is to provide an interoperable means for devices from different manufacturers to upgrade each other's image. Additionally, the OTA Upgrade cluster defines a mechanism by which security credentials, logs and configuration file types are accessible by offering a solution that utilizes a set of optional and mandatory commands.

11.1.2 Scope

This chapter will only describe features that require implementation in order to be ZigBee OTA upgrade (cluster) certified. Other optional features including using multicast for sending upgrade messages and (upgrade) cloning will not be discussed in this document.

Currently, only Application Bootloader support is required in order to support ZigBee OTA Upgrade cluster. MAC Bootloader upgrading is not supported at the moment.

11.1.3 Terminology

Application Standard or Standard – This is a noun that refers to any application standard specification that includes this specification. Examples: ZigBee Home Automation, ZigBee Smart Energy, etc.

11.2 General Description

11.2.1 Introduction

The existing OTA upgrade methods available are platform specific, not OTA interoperable and do not provide a common framework for upgrading networks that support a mix of devices from multiple platforms and ZigBee Stack vendors.

The intent of this chapter is to provide an interoperable OTA upgrade of new image for devices deployed in the field. As long as the device supports the OTA Upgrade cluster and it is certified by an approved test house, its image SHALL be upgradeable by another device from the same or different manufacturer that also implemented and certified the OTA Upgrade cluster.

OTA Upgrade cluster will also require that in order to support OTA upgrade, the device will need to have an application bootloader installed as well as sufficient memory (external or internal) to store the newly loaded image. An application bootloader uses the running ZigBee stack and application to retrieve and store a new image. Depending upon the manufacturer, the image MAY consist of a bootloader image, a ZigBee stack image or only a patch to the application image. Whatever comprises the OTA upgrade image being sent to the node does not concern the ZigBee OTA cluster and it is outside the scope.

To use an application bootloader, the device is required to have sufficient memory (internal or external) to store the newly downloaded OTA upgrade image. By doing so, the current running image is not overwritten until the new image has been successfully downloaded. It also allows the possibility of a node saving an image in its memory and forwarding that image to another node. Application bootloading provides flexibility of when the device decides to download new OTA upgrade image as well as when the device decides to switch to running the new image.

Since the bootloading is done at the application level, it automatically makes use of various features already offered by the ZigBee Network Layer and Application Sub Layer (APS) including the ability to bootload a device that is multiple hops away, message retries to increase reliability, and security. It also allows the network to continue to operate normally while the bootload is in progress. In addition, it supports bootloading of sleeping (RxOnWhenIdle=FALSE) devices.

The application bootload messages are built upon typical ZigBee messages, with additional ZigBee Cluster Library (ZCL) header and payload and ZigBee OTA cluster specific payload.

An application standard that includes this specification MAY, of course, add requirements and dependencies not defined here. Such a standard SHALL not relax requirements by changing a feature here from mandatory to optional, but it MAY specify features it deems mandatory, that are optional in this specification.

For example: The ZigBee Smart Energy standard has particular OTA cluster security feature requirements that are defined as mandatory in the ZSE specification, but are optional here.

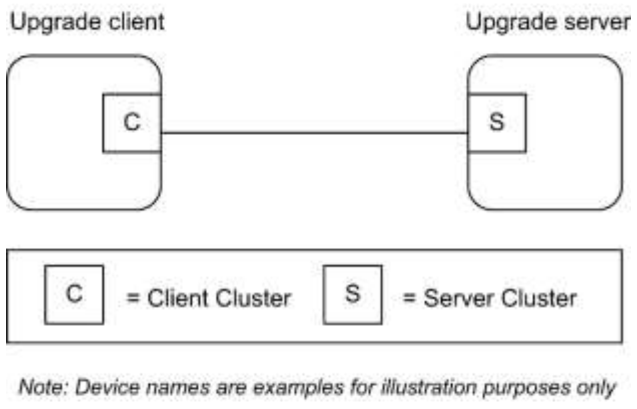
11.2.2 Cluster List

The clusters defined in this document are listed in Table 11-1.

Table 11-1. Clusters Specified in This Document

Cluster ID	Cluster Name	Description
0x0019	OTA Upgrade	Parameters and commands for upgrading image on devices Over-The-Air.

Figure 11-1. Typical Usage of OTA Upgrade Cluster



Upgrade Client is a device to be upgraded with new image. Upgrade server is a device that has the new image to send to the client. This document SHALL specify how the client discovers the server, the over the air message format between the client and server, and the means for the server to signal the client to switch to running the new image.

It is possible that the upgrade server MAY have several OTA upgrade images from different manufacturers. How the upgrade server receives these OTA upgrade images and how it stores and manages them are outside the scope of this document.

In addition to the typical use case of transferring new firmware images to client devices, OTA Upgrade cluster MAY also be used to transfer device specific file types such as log, configuration or security credentials (needed for upgrading from SE 1.0 or SE 1.1 to SE 2.0). The cluster provides flexibility in OTA header and a set of optional commands that make transferring of such file types possible.

11.3 OTA Upgrade

11.3.1 Overview

Please see section 2.2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

The cluster provides a standard way to upgrade devices in the network via OTA messages. Thus the upgrade process MAY be performed between two devices from different manufacturers. Devices are required to have application bootloader and additional memory space in order to successfully implement the cluster.

It is the responsibility of the server to indicate to the clients when update images are available. The client MAY be upgraded or downgraded. The upgrade server knows which client devices to upgrade and to what file version. The upgrade server MAY be notified of such information via the backend system. For ZR clients, the server MAY send a message to notify the device when an updated image is available. It is assumed that ZED clients will not be awake to receive an unsolicited notification of an available image. All clients (ZR and ZED) SHALL query (poll) the server periodically to determine whether the server has an image update for them. Image Notify is optional.

The cluster is implemented in such a way that the client service works on both ZED and ZR devices. Being able to handle polling is mandatory for all server devices while being able to send a notify is optional. Hence, all client devices must be able to use a ‘poll’ mechanism to send query message to the server in order to see if the server has any new file for it. The polling mechanism also puts fewer resources on the upgrade server. It is ideal to have the server maintain as little state as possible since this will scale when there are hundreds of clients in the network. The upgrade server is not required to keep track of what pieces of an image that a particular client has received; instead the client SHALL do that. Lastly poll makes more sense for devices that MAY need to perform special setup to get ready to receive an image, such as unlocking flash or allocating space for the new image.

11.3.1.1 Revision History

The global mandatory *ClusterRevision* attribute SHALL reflect the revision of the implemented cluster specification as identified by one or more cluster identifiers listed below in this specification (see 2.3.5.1).

Rev	Description
1	mandatory global *ClusterRevision* attribute added; CCBs 1374 1470 1477 1540 1594 2046 2056
2	alternative Image Activation Policies; 128-bit Crypto suite, Smart Energy Profile 1.2a & 1.2b
3	CCB 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2296 2307 2315 2342 2398 2464

11.3.1.2 Classification

Hierarchy	Role	PICS Code

Base	Utility[124]	OTA

11.3.1.3 Cluster Identifiers

Identifier	PICS Code	Name
0x0019	OTA	OTA Upgrade

11.3.2 Security

Security for the OTA Upgrade cluster encompasses these areas: image verification, image transport, and image encryption. Security mechanisms in the application standard dictate the security level of over-the-air image upgrading. For example, an application standard with strict security policies (such as Smart Energy) MAY support image signature as well as encryption in both network and APS layers; while other application standards MAY only support network encryption. Each application standard must decide the list of required security policies for their use of the OTA Upgrade cluster.

11.3.2.1 Terminology

There are many aspects to security. These can be broken down into the following areas: confidentiality, integrity, authentication, availability, and non-repudiation. Authorization and auditing can also be considered as part of security.

Confidentiality – This is the requirement that no third party can read data that is not intended for that party. This is discussed below in Image Encryption.

Integrity – This is the property of security where the recipient can verify that data was not modified between the time it was initially distributed by the sender and when it was received by the intended recipient. Modifications to the data by third parties can be detected. This is discussed in Image Verification below.

Authentication – This is the property where the identity of the sender of data can be verified by the intended recipient. This is discussed in Image Verification below.

Availability – This refers to the property that resources are available when they are required and cannot be unfairly consumed by an attacker. The OTA Upgrade cluster does not address this; as such it will not be discussed any further.

Non-repudiation – This refers to the property where a sender/receiver cannot deny that a security exchange took place. The OTA Upgrade cluster does not address this; as such it will not be discussed any further.

11.3.3 Image Verification

11.3.3.1 Asymmetric Verification of Authenticity and Integrity

It is strongly encouraged that there is a means to verify the authenticity and integrity of the bootload image. This is most often accomplished through asymmetric encryption technologies (i.e. public/private keys) where only one device is able to create a digital signature but many devices are able to verify it. Bootload images MAY be signed by the private key of the manufacturer with that signature appended to the image that is transported to the device. Once the complete image has been received the signature is verified using the public key of the signer.

Devices MAY be pre-installed with the certificate (public key) of the device that created the signature, or they MAY receive the certificate over-the-air. How the signer's security data is obtained is considered outside the scope of the OTA Upgrade cluster and is manufacturer specific. When signer certificates are sent over-the-air and not pre-installed, it is recommended that the transportation of the certificate be done using encryption from a trusted source to reduce the chance an attacker MAY inject their own signer certificate into the device.

[124] CCB 2315 OTA is a Utility, not an Application class cluster

Images with verification mechanisms built in MAY be transported over insecure communication mechanisms while still maintaining their authenticity and integrity. In fact, it is likely that the originator of the upgrade image (the manufacturer) will not be directly connected to any ZigBee networks and therefore distribute the upgrade image across other mediums (such as the internet) before arriving on the ZigBee network. In that case it is crucial that the Image Verification be independent of the communication medium. Any attempts to tamper with the signature or the data itself must be detected and will cause the upgrade image to be rejected by the target device. An attacker that crafts its own signed image and tries to have it accepted will be rejected since that image will not be signed by the manufacturer's signing authority.

It is up to each application standard to determine the minimum requirements for image verification. For example: for the ZigBee Smart Energy standard there is already a minimum set of security requirements included in NEMA SG-AMI 1-2009. The OTA Upgrade cluster will communicate the methods in use for basic image verification. Individual manufacturers are free to augment this and provide their own extensions. Those extensions are outside the scope of the OTA Upgrade cluster.

Without asymmetric encryption technology, a device is limited in its ability to authenticate images. Images MAY be encrypted with symmetric keys such that only those devices that need to decrypt the image have access to the key. However, the security of this system is dependent on the security of all devices that have access to the symmetric key.

11.3.3.2 Verification of Integrity by Hash Value

For application standards that do not require the asymmetric verification method, the authenticity of the OTA image cannot be verified. However, it is possible to verify the integrity of the OTA image by using the hash value method in section 11.7.2. There will be no signer certificate nor any signature involved.

11.3.4 Image Transport

When there is a means to verify the authenticity of the bootload images, transport of the images in a secure fashion provides little additional security to the integrity and authenticity. What secure transportation provides is a means to communicate the policies about *when* a device SHOULD perform upgrade or *what version* it SHOULD upgrade to.

A secured ZigBee network uses network security for all messages, but that does not provide point-to-point security. APS security SHOULD be used to assure that messages are sent from only the trusted source (the upgrade server). This will be utilized to provide implicit and explicit authorization by the upgrade server about when devices will *initiate* bootload events.

Distribution of upgrade image via broadcast or multicast messages is not recommended because its lack of point-to-point security. Reception of upgrade images via broadcast or multicast SHOULD not be inferred as authorization by the upgrade server to initiate the upgrade. In this case, the act of receiving the image and upgrading it SHOULD be split up into separate events. The latter communications SHOULD be done via unicast to verify that the upgrade server has authorized a device to upgrade to a previously received image. Applications must determine what level of authorization is required by the upgrade server.

11.3.5 Image Signature

An application standard MAY require that the OTA Upgrade cluster provides mechanisms to sign the OTA file to protect the authenticity and integrity of the image.

11.3.6 Image Integrity Code

An application standard MAY require that the OTA Upgrade cluster provides hash mechanisms to provide protection against unintended data corruption.

11.4 OTA File Format

11.4.1 General Structure

The OTA file format is composed of a header followed by a number of sub-elements. The header describes general information about the file such as version, the manufacturer that created it, and the device it is intended for. Sub-elements in the file MAY contain upgrade data for the embedded device, certificates, configuration data, log messages, or other manufacturer specific pieces. Below is an example file.

Figure 11-2. Sample OTA File

Octets	**Variable**	**Variable**	**Variable**	**Variable**
Data	OTA Header	Upgrade Image	Signer Certificate	Signature

The OTA header will not describe details of the particular sub-elements. Each sub-element is self-describing. With exception of a few sub-elements, the interpretation of the data contained is up to the manufacturer of the device.

11.4.2 OTA Header Format

Table 11-2. OTA Header Fields

Octets	Data Types	Field Names	M/O
4	uint32	OTA upgrade file identifier	M
2	uint16	OTA Header version	M
2	uint16	OTA Header length	M
2	map16[125]	OTA Header Field control	M
2	uint16	Manufacturer code	M
2	uint16	Image type	M
4	uint32	File version	M
2	uint16	ZigBee Stack version	M
32	ASCII[126]	OTA Header string	M
4	uint32	Total Image size (including header)	M
0/1	uint8	Security credential version	O
0/8	EUI64	Upgrade file destination	O
0/2	uint16	Minimum hardware version	O
0/2	uint16	Maximum hardware version	O

[125] CCB 2219 Field Control is 16-bit bitmap, not an unsigned 16-bit integer
[126] this does not have a length byte, so it is not a character string data type

The first entry of the table above (OTA upgrade file identifier) represents the first field in the OTA header, and the last entry represents the last field. The endianness used in each data field SHALL be little endian in order to be compliant with general ZigBee messages.

Please refer to Chapter 2, Foundation Specification, for more description on data types.

11.4.2.1 OTA Upgrade File Identifier

The value is a unique 4-byte value that is included at the beginning of all ZigBee OTA upgrade image files in order to quickly identify and distinguish the file as being a ZigBee OTA cluster upgrade file, without having to examine the whole file content. This helps distinguishing the file from other file types on disk. The value is defined to be "0x0BEEF11E".

11.4.2.2 OTA Header Version

The value enumerates the version of the header and provides compatibility information. The value is composed of a major and minor version number (one byte each). The high byte (or the most significant byte) represents the major version and the low byte (or the least significant byte) represents the minor version number. A change to the minor version means the OTA upgrade file format is still backward compatible, while a change to the major version suggests incompatibility.

The current OTA header version SHALL be 0x0100 with major version of "01" and minor version of "00".

11.4.2.3 OTA Header Length

This value indicates full length of the OTA header in bytes, including the OTA upgrade file identifier, OTA header length itself to any optional fields. The value insulates existing software against new fields that MAY be added to the header. If new header fields added are not compatible with current running software, the implementations SHOULD process all fields they understand and then skip over any remaining bytes in the header to process the image or signing certificate. The value of the header length depends on the value of the OTA header field control, which dictates which optional OTA header fields are included.

11.4.2.4 OTA Header Field Control

The bit mask indicates whether additional information such as Image Signature or Signing Certificate are included as part of the OTA Upgrade Image.

Table 11-3. OTA Header Field Control Bitmask

Bits	Name
0	Security Credential Version Present
1	Device Specific File
2	Hardware Versions Present

Security credential version present bit indicates whether security credential version field is present or not in the OTA header.

Device specific file bit in the field control indicates that this particular OTA upgrade file is specific to a single device.

Hardware version present bit indicates whether minimum and maximum hardware version fields are present in the OTA header or not.

11.4.2.5 Manufacturer Code

This is the ZigBee assigned identifier for each member company. When used during the OTA upgrade process, manufacturer code value of 0xffff has a special meaning of a wild card. The value has a 'match all' effect. OTA server MAY send a command with wild card value for manufacturer code to match all client devices from all manufacturers.

11.4.2.6 Image Type

The manufacturer SHOULD assign an appropriate and unique image type value to each of its devices in order to distinguish the products. This is a manufacturer specific value. However, the OTA Upgrade cluster has reserved the last 64 values of image type value to indicate specific file types such as security credential, log, and configuration. When a client wants to request one of these specific file types, it SHALL use one of the reserved image type values instead of its own (manufacturer specific) value when requesting the image via Query Next Image Request command.

Table 11-4. Image Type Values

File Type Values	File Type Description
0x0000 – 0xffbf	Manufacturer Specific
0xffc0	Client Security credentials
0xffc1	Client Configuration
0xffc2	Server Log
0xffc3	Picture
0xffff	wild card

Image type value of 0xffff has a special meaning of a wild card. The value has a 'match all' effect. For example, the OTA server MAY send Image Notify command with image type value of 0xffff to indicate to a group of client devices that it has all types of images for the clients. Additionally, the OTA server MAY send Upgrade End Response command with image type value of 0xffff to indicate a group of clients, with disregard to their image types, to upgrade.

11.4.2.7 File Version

For firmware image, the file version represents the release and build number of the image's application and stack. The application release and build numbers are manufacturer specific, however, each manufacturer SHOULD obtain stack release and build numbers from their stack vendor. OTA Upgrade cluster makes the recommendation below regarding how the file version SHOULD be defined, in an attempt to make it easy for humans and upgrade servers to determine which versions are newer than others. The upgrade server SHOULD use this version value to compare with the one received from the client.

The server MAY implement more sophisticated policies to determine whether to upgrade the client based on the file version. A higher file version number indicates a newer file.

Table 11-5. Recommended File Version Definition

Application Release	Application Build	Stack Release	Stack Build
1 byte	1 byte	1 byte	1 byte
8-bit integer	8-bit integer	8-bit integer	8-bit integer

For example,

File version A: 0x10053519 represents application release 1.0 build 05 with stack release 3.5 b19.

File version B: 0x10103519 represents application release 1.0 build 10 with stack release 3.5 b19.

File version C: 0x10103701 represents application release 1.0 build 10 with stack release 3.7 b01.

File version B is newer than File version A because its application version is higher, while File version C is newer than File version B because its stack version is higher.

The file version value MAY be defined differently for different image types. For example, version scheme for security credential data MAY be different than that of log or configuration file or a normal firmware upgrade image version. The specific implementation of a versioning scheme is manufacturer specific.

Note that a binary-coded decimal convention (BCD) concept is used here for version number. This is to allow easy conversion to decimal digits for printing or display, and allows faster decimal calculations.

11.4.2.8 ZigBee Stack Version

This information indicates the ZigBee stack version that is used by the application. This provides the upgrade server an ability to coordinate the distribution of images to devices when the upgrades will cause a major jump that usually breaks the over-the-air compatibility, for example, from ZigBee Pro to upcoming ZigBee IP. The values below represent currently available ZigBee stack versions.

Table 11-6. ZigBee Stack Version Values

ZigBee Stack Version Values	Stack Name
0x0000	ZigBee 2006
0x0001	ZigBee 2007
0x0002	ZigBee Pro
0x0003	ZigBee IP

11.4.2.9 OTA Header String

This is a manufacturer specific string that MAY be used to store other necessary information as seen fit by each manufacturer. The string SHALL be a null terminated string using ASCII encoding. Any bytes after the terminating character MAY be used by manufacturers for additional data transport and SHALL not be interpreted as human readable data. The idea is to have a human readable string that can prove helpful during development cycle. The string is defined to occupy 32 bytes of space in the OTA header.

11.4.2.10 Total Image Size

The value represents the total image size in bytes. This is the total of data in bytes that SHALL be transferred over-the-air from the server to the client. In most cases, the total image size of an OTA upgrade image file is the sum of the OTA header and the actual file data (along with its tag) lengths. If the image is a signed image and contains a certificate of the signer, then the Total image size SHALL also include the signer certificate and the signature (along with their tags) in bytes.

This value is crucial in the OTA upgrade process. It allows the client to determine how many image request commands to send to the server to complete the upgrade process.

11.4.2.11 Security Credential Version

This information indicates security credential version type, such as SE1.0 or SE2.0 that the client is required to have, before it SHALL install the image. One use case for this is so that after the client has downloaded a new image from the server, it SHOULD check if the value of security credential version allows for running the image. If the client's existing security credential version does not match or is outdated from what specified in the OTA header, it SHOULD obtain new security credentials before upgrading to running the new image.

Table 11-7. Security Credential Version

Security Credential Version Values	Security Credential Version Types
0x00	SE 1.0
0x01	SE 1.1
0x02	SE 2.0
0x03	SE 1.2

11.4.2.12 Upgrade File Destination

If Device Specific File bit is set, it indicates that this OTA file contains security credential/certificate data or other type of information that is specific to a particular device. Hence, the upgrade file destination field (in OTA header) SHOULD also be set to indicate the IEEE address of the client device that this file is meant for.

11.4.2.13 Minimum Hardware Version

The value represents the earliest hardware platform version this image SHOULD be used on. This field is defined as follows:

Table 11-8. Hardware Version Format

Version	Revision
1 byte	1 byte
8-bit integer	8-bit integer

The high byte represents the version and the low byte represents the revision.

11.4.2.14 Maximum Hardware Version

The value represents the latest hardware platform this image SHOULD be used on. The field is defined the same as the Minimum Hardware Version (above).

The hardware version of the device SHOULD not be earlier than the minimum (hardware) version and SHOULD not be later than the maximum (hardware) version in order to run the OTA upgrade file.

11.4.3 Sub-element Format

Sub-elements in the file are composed of an identifier followed by a length field, followed by the data. The identifier provides for forward and backward compatibility as new sub-elements are introduced. Existing devices that do not understand newer sub-elements MAY ignore the data.

Figure 11-3. Sub-element Format

Octets	**2-bytes**	**4-bytes**	**Variable**
Data	Tag ID	Length Field	Data

Sub-elements provide a mechanism to denote separate sections of data utilized by the device for the upgrade. For example, a device that has multiple processors each with their own firmware image could use a separate sub-element for each one. The details of how this is handled would be up to the manufacturer of the device.

A few sub-elements are not manufacturer-specific and defined by the OTA cluster itself. See section 11.4.4 below.

11.4.3.1 Tag ID

The tag identifier denotes the type and format of the data contained within the sub-element. The identifier is one of the values from Table 11-9 below.

11.4.3.2 Length Field

This value dictates the length of the rest of the data within the sub-element in bytes. It does not include the size of the Tag ID or the Length Fields.

11.4.3.3 Data

The length of the data in the sub-element must be equal to the value of the Length Field in bytes. The type and format of the data contained in the sub-element is specific to the Tag.

11.4.4 Tag Identifiers

Sub-elements are generally specific to the manufacturer and the implementation. However, this specification has defined a number of common identifiers that MAY be used across multiple manufacturers.

Table 11-9. Tag Identifiers

Tag Identifiers	**Description**
0x0000	Upgrade Image
0x0001	ECDSA Signature (Crypto Suite 1)
0x0002	ECDSA Signing Certificate (Crypto Suite 1)
0x0003	Image Integrity Code
0x0004	Picture Data
0x0005	ECDSA Signature (Crypto Suite 2)

Tag Identifiers	Description
0x0006	ECDSA Signing Certificate (Crypto Suite 2)
0xf000 – 0xffff	Manufacturer Specific Use

Manufacturers MAY define tag identifiers for their own use and dictate the format and behavior of devices that receive images with that data.

11.4.5 Crypto Suites

The specification allows use of one of two crypto suites. Crypto Suite 1 corresponds to the version offering ‘80-bit’ symmetric equivalent security, Crypto Suite 2 allows ‘128-bit’ symmetric equivalent security. Each utilizes different key lengths and signature sizes and thus requires unique tags with different sizes.

11.4.6 ECDSA Signature Sub-element (Crypto Suite 1)

The ECDSA Signature sub-element contains a signature for the entire file as means of insuring that the data was not modified at any point during its transmission from the signing device.

If an image contains an ECDSA Signature Sub-element it SHALL be the last sub-element in the file.

Figure 11-4. ECDSA Signature

Octets	2-bytes	4-bytes	8-bytes	42-bytes
Data	Tag ID: 0x0001	Length Field: 0x00000032	Signer IEEE Address	Signature Data

11.4.6.1 Signer IEEE Address

This field SHALL contain the IEEE address of the device that created the signature, in little endian format.

11.4.6.2 Signature Data

This field SHALL contain the ECDSA signature data, and is generated as described in the section 11.7.

11.4.7 ECDSA Signing Certificate Sub-element

This sub-element is used to include information about the authority that generated the signature for the OTA file.

Figure 11-5. ECDSA Signing Certificate Sub-element

Octets	2-bytes	4-bytes	48-bytes
Data	Tag ID: 0x0002	Length Field: 0x00000030	ECDSA Certificate

11.4.7.1 ECDSA Certificate (Crypto Suite 1)

This SHALL contain the data for the ECDSA certificate of the device. The certificate SHALL be formatted as described in [Z9] in section C.4.2.2.4.

11.4.8 Image Integrity Code Sub-element

This sub-element includes a hash value used to verify the integrity of the OTA file.

Figure 11-6. Hash Value Sub-element

Octets	2-bytes	4-bytes	16-bytes
Data	Tag ID: 0x0003	Length Field: 0x00000010	Hash Value

11.4.8.1 Hash Value

This hash value used to verify the integrity of the OTA file and the detail to generate the hash is listed in section 11.7.2.

11.4.9 ECDSA Signature Sub-element (Crypto Suite 2)

The ECDSA Signature sub-element contains a signature for the entire file as means of insuring that the data was not modified at any point during its transmission from the signing device.

If an image contains an ECDSA Signature Sub-element it SHALL be the last sub-element in the file.

Figure 11-7. ECDSA Signature

Octets	2-bytes	4-bytes	8-bytes	72-bytes
Data	Tag ID: 0x0005	Length Field: 0x00000050	Signer IEEE Address	Signature Data

11.4.9.1 Signer IEEE Address

This field SHALL contain the IEEE address of the device that created the signature, in little endian format.

11.4.9.2 Signature Data

This field SHALL contain the ECDSA signature data, and is generated as described in the section 11.7.

11.4.10 ECDSA Signing Certificate Sub-element (Crypto Suite 2)

This sub-element is used to include information about the authority that generated the signature for the OTA file.

Figure 11-8. ECDSA Signing Certificate Sub-element

Octets	**2-bytes**	**4-bytes**	**74-bytes**
Data	Tag ID: 0x0006	Length Field: 0x0000004A	ECDSA Certificate

11.4.10.1 ECDSA Certificate (Crypto Suite 2)

This SHALL contain the data for the ECDSA certificate of the device. The certificate SHALL be formatted as described in [Z9] in section C.4.2.3.3.

11.5 OTA File Naming

OTA Upgrade cluster provides recommendation below regarding OTA Upgrade image file naming convention and extension. This is an effort to assist the upgrade server in sorting different image files received from different manufacturers.

The OTA Upgrade image file name SHOULD contain the following information at the beginning of the name with each field separated by a dash ("-"): manufacturer code, image type and file version. The value of each field stated SHOULD be in hexadecimal number and in capital letter. Each manufacturer MAY append more information to the name as seen fit to make the name more specific. The OTA Upgrade file extension SHOULD be ".zigbee".

An example of OTA Upgrade image file name and extension is "1001-00AB-10053519-upgradeMe.zigbee".

11.6 Signatures

It is up to the application standard to determine whether or not a signature is necessary for over the air upgrade files. If a standard has mandated the use of signatures then a device adhering to that standard SHALL only accept images that have a signature sub-element. If such a device receives an OTA file that does not contain a signature sub-element, then the device will discard the image and proceed with any further processing required by the specific application standard. The device must verify the signature as described in the following sections prior to acting on any data inside the file.

If a standard does not require the use of signatures then devices MAY still choose to use images with signatures. However, it is highly recommended that such a device only accept images either with signatures or without, but not accept both. A device greatly reduces its security if it will accept signed or unsigned upgrade files.

11.7 ECDSA Signature Calculation

It is EXPECTED that in most all cases the signer device is not a real ZigBee device and is not part of any ZigBee network. Therefore, the signer's IEEE is not a real ZigBee device address, but the address of a virtual device that exists only to sign upgrade images for a manufacturer and or a set of products. Its address SHOULD be separate from the block of device addresses produced by a manufacturer as certified ZigBee devices.

The signature calculation SHALL be performed as follows:

1. A valid OTA image SHALL have previously been created including all the necessary header fields, tags, and their data, in the image.
2. The signer shall select a crypto suite to use based on its own security policies.
3. An ECDSA signer certificate tag sub-element SHALL be constructed, based on the selected crypto suite, and SHALL contain the certificate of the signing device, appended to the image.
4. An ECDSA signature tag sub-element SHALL be constructed, based on the previously selected crypto suite, including only the tag ID, the length of the tag, and the signer's IEEE address (see 11.4.6 and 11.4.9). No actual signature data SHALL be included yet. The tag SHALL be appended to the image.
5. The OTA image header SHALL be updated with a new total image size, including the signature certificate tag sub-element that was added, and the full size of the ECDSA signature tag sub-element (see 11.4.6 and 11.4.9).
6. A message digest SHALL be calculated over the entire image.
    a. The message digest SHALL be computed by using the Matyas-Meyer-Oseas cryptographic hash specified in the ZigBee core specification 05-3474-20 Section B.6. This uses the extended AES-MMO hash proposed as a change to an earlier version of the ZigBee core specification 05-3474-20 Section B.6.

2. The ECDSA algorithm SHALL be used to calculate the signature using the message digest and the signer device's private key.
3. The r and s components of the signature SHALL both be appended to the image. The r component SHALL be appended first, and then the s component.

11.7.1 ECDSA Signature Verification

The signature of a completely downloaded OTA file SHALL be verified as follows.

1. The ZigBee device SHALL first determine if the signer of the image is an authorized signer.
    a. It does this by extracting the signer IEEE from ECDSA signature tag sub-element.
        a. If an ECDSA signature tag sub-element is not found in the image, then the image SHALL be discarded as invalid and no further processing SHALL be done.
    b. The device SHALL compare the extracted signer IEEE with the list of local, known, authorized signers and determine if there is a match.
    c. If no match is found, then the image SHALL be discarded as invalid and no further processing SHALL be done.
2. The device SHALL then check the ECDSA Crypto Suite of the image.
    a. The device SHALL check which crypto suite is used for both the ECDSA Signature tag and ECDSA Signing Certificate tag. If both elements do not use the same crypto suite, then the device SHALL discard the image as invalid and no further processing SHALL be done.
    b. The device SHALL check which crypto suites are locally allowed and supported. If the device does not locally support the crypto suite used by the ECDSA Signing certificate tag or a security policy does not allow its use for verifying locally, then it SHALL discard the image as invalid and no further processing SHALL be done.
3. The device SHALL then obtain the certificate associated with the signer IEEE.
    a. The device SHALL extract the signer certificate data from the ECDSA signing certificate sub-element.

    If there is no ECDSA signing certificate tag sub-element, then it SHALL discard the image as invalid and no further processing SHALL be done.

   b. The device SHALL verify that the signer IEEE address within the ECDSA signature tag sub-element matches the subject field of the ECDSA signing certificate sub-element.

      Note: The subject field IEEE is in big-endian format and the signer IEEE is in little endian format.

   c. If the addresses do not match, then the image SHALL be discarded as invalid and no further processing SHALL be done.

4. The device SHALL then obtain the CA public key associated with the signer.
   a. The device SHALL obtain the IEEE of the CA public key from the issuer field within the ECDSA certificate data of the ECDSA signing certificate sub-element.
   b. If the IEEE of the CA does not match its list of known CAs, or the public key for that CA could not be locally obtained, then the image SHALL be discarded as invalid and no further processing SHALL be done.
5. The device SHALL then calculate the message digest of the image.

   The digest SHALL be calculated using the Matyas-Meyer-Oseas cryptographic hash function over the entire image except for the signature data of the ECDSA signature sub-element.

   Note: The calculation SHALL include the signature tag ID of the ECDSA signature sub-element, the length field of the ECDSA signature sub-element, and the signer IEEE field of the ECDSA signature sub-element.
6. The signer’s public key SHALL be obtained by extracting it from the signer certificate.
7. The device SHALL then pass the calculated digest value, signer certificate, and CA public key to the ECDSA verification algorithm.
8. If the ECDSA algorithm returns success, then the image SHALL be considered valid.
9. If the ECDSA algorithm returns any other result, then the image SHALL be discarded as invalid and no further processing SHALL be done.

11.7.2 Image Integrity Code

It is up to the application standard to determine whether or not an image integrity code is necessary for over the air upgrade files. Standards that require the use of digital signatures SHALL NOT use this Image Integrity Hash Code sub-element in conjunction to the ECDSA signature. If a standard has mandated the use of hash values then a device adhering to that standard SHALL only accept images that have a valid hash sub-element. If such a device receives an OTA file that does not contain a hash sub-element, then the device will discard the image and proceed with any further processing required by the specific application standard. The device must verify the hash as described in the following sections prior to acting on any data inside the file.

The hash value provides protection against unintended data corruption. An OTA image which is hosted at a back-end image repository might be stored and forwarded at several intermediate locations before it reaches the OTA server, where it is typically stored on a local file system. There is a potential for this file being corrupted either during transfer or as a result of file system errors. The hash value provides an interoperable way for OTA servers to detect corrupt images before advertising such files to OTA clients. Otherwise corrupt images might only be detected by OTA clients after complete download over-the-air. Since this condition cannot be detected by the OTA server, it would offer the same (corrupt) file over and over again.

If the application standard does not mandate the use of this hash value, it is strongly recommended that image integrity is ascertained using another approach, for example a hash value (SHA-256 or comparable) that is maintained out-of-band and provided by the device vendor together with the OTA image.

11.7.2.1 Hash Value Calculation

The hash value calculation SHALL be performed as follows:

1. A valid OTA image SHALL have previously been created including all the necessary header fields, tags, and their data, in the image.
2. An AES-MMO hash value tag sub-element header SHALL be constructed including only the tag ID and the length of the sub-element data (16 bytes). No actual data SHALL be included yet. The tag header SHALL be appended to the image.
3. The OTA image header SHALL be updated with a new total image size, including the hash tag sub-element header that was added, and the full size of the hash tag sub-element (6 + 16 = 22 bytes).
4. The hash value SHALL be calculated using the Matyas-Meyer-Oseas cryptographic hash specified in section B.6 of [R5]. The hash is calculated starting with the OTA image header and spanning just before the hash sub-element header, i.e. the calculation takes in to account the first byte of the image header up to the last byte of the sub-element preceding the hash sub-element.
5. The computed hash value SHALL be appended to the image.

11.7.2.2 Hash Value Verification

The hash value of a complete OTA file SHALL be verified as follows.

1. The device SHALL calculate the hash value of the image using the Matyas-Meyer-Oseas cryptographic hash function. The hash is calculated starting with the OTA image header and spanning just before the hash sub-element header, i.e. the calculation takes in to account the first byte of the image header up to the last byte of the sub-element preceding the hash sub-element.
2. The device SHALL then compare the calculated hash value with the data stored in the hash tag sub-element data. If both octet strings are equal, the image SHALL be considered intact, otherwise the image SHALL be considered corrupt.
3. If the image is regarded corrupt, it SHALL be discarded as invalid and no further processing SHALL be done.

11.8 Discovery of the Upgrade Server

Before becoming part of the network, a device MAY be preprogrammed with the IEEE address of the authorized upgrade server. In this case, once the device is part of the network, it SHALL discover the network address of the upgrade server via ZDO network address discovery command.

If the device is not preprogrammed with the upgrade server's IEEE address, the device SHALL discover the upgrade server before it participates in any upgrade process. The device SHALL send Match Descriptor Request (ZDO command) to discover an upgrade server by specifying a single OTA cluster ID in the input Cluster attribute. If the receiving node is an upgrade server, it SHALL reply with Match Descriptor Response, with the (active) endpoint that the OTA cluster is implemented on, hence, identifying itself as acting as server in OTA Upgrade cluster. Since Match descriptor request MAY be sent as unicast or broadcast, the client MAY get multiple responses if there are more than one server in the network. The client SHALL use the first response received. Each application standard SHOULD specify the frequency of OTA server discovery done by the client. After discovering the OTA server's short ID via the ZDO Match descriptor request the client SHALL discover the IEEE address of the upgrade server via ZDO IEEE address discovery command and store the value in UpgradeServerID attribute.

A node SHALL have an application link key with the Upgrade server; it SHALL request one prior to any OTA operations.

If the upgrade server is the trust center, it SHOULD use its trust center link key.

In the case of the ZigBee Smart Energy standard, where the upgrade server is not the trust center, the device SHALL perform partner link key request.

11.8.1 Server and Client

The server must be able to store one or more OTA upgrade image(s). The server MAY notify devices in the network when it receives new OTA upgrade image by sending an Image Notify Command. The Image Notify Command will be received reliably only on ZR devices since ZED devices MAY have their radio off at the time. The Image Notify Command MAY be sent as unicast or broadcast. If sent as broadcast, the message also has a jitter mechanism built in to avoid the server being overwhelmed by the requests from the clients. If sent as unicast, the client SHALL ignore the jitter value.

The client device will send Query Next Image Request Command if the information in the Image Notify Command is of interest and after applying the jitter value. All devices SHALL send in a Query Next Image Request Command periodically regardless of whether an Image Notify was sent by the OTA server.

When the device has received a response to its query indicating a new OTA upgrade image is available, the client device SHALL request blocks of the OTA upgrade image. The process continues until the client receives all image data. At that point, the client SHALL verify the integrity of the whole image received and send Upgrade End Request Command along with the upgrade status. The server SHALL notify the client of when to upgrade to new image in the Upgrade End Response.

It is the responsibility of the server to ensure that all clients in the network are upgraded. The server MAY be told which client to upgrade or it MAY keep a database of all clients in the network and track which client has not yet been upgraded.

11.8.2 Sleepy Devices

The upgrade server has no reliable way to immediately notify the sleepy devices of the availability of new OTA Upgrade image, hence, the devices SHALL query the server periodically to learn if there are new images available. The query for new upgrade image MAY be done as a separate event or it MAY be done in addition to normal scheduled communication between the device and the server. The frequency as to how often the sleepy devices query the server SHALL be specified by each application standard. Moreover, it is important to realize that the frequency that the sleepy device checks for new image (sending Query Next Image command) determines how often the particular node could be upgraded. This rate will also drive how fast code updates MAY be pushed out to each network. For the SE 1.x to SE 2.0 transition, if sleepy devices only check in once a month for the new image then it will likely to take over a month to complete the transition. If the application standard fails to set any requirement on the sleepy device checking for new images, then it is unlikely that the OTA upgrade feature will work reliably for those devices.

It is a recommendation that sleepy devices SHALL make their best effort to poll more rapidly during the OTA Upgrade Image download process in order to ensure that the download completes in a timely manner. However, it is acknowledged that some sleepy devices MAY not be able to do so due to limitation on their batteries or due to other reasons such as battery-less/Green Power devices. Hence, such devices MAY take much longer to complete the download process.

11.9 Dependencies

Each device that wishes to implement the OTA Upgrade cluster SHALL have the following:

- ZigBee Device Object (ZDO) match descriptor request and response commands. The command is used to discover upgrade server.
- ZigBee Cluster Library (ZCL) global commands and basic cluster attributes.
- Application Bootloader: To actually upgrade existing image with newly installed one on the additional memory space. The implementation of the Bootloader along with its specification, for example, where it lives and its size are outside the scope of this document.
- Additional Memory Space SHALL be large enough to hold the whole OTA Upgrade Image: It is important to be able to store the new image until the device receives a signal from the server to switch to running the image.

This is because it MAY be necessary for all devices in the network to switch their images at once if the new image is not OTA compatible with the old one.

- In addition, if the client device is composed of multiple processors; each requires separate image, then the additional memory space SHALL be large enough to hold all the images for all the processors that make up the device. In case of server devices, its additional memory space will depend on how many images the devices are planning to hold.
- The specification of the additional memory space and its connection to the processor is outside the scope of this document.

11.10 OTA Cluster Attributes

Below are attributes defined for OTA Upgrade cluster. Currently, **all attributes are client side attributes** (only stored on the client). There is no server side attribute at the moment. All attributes with the exception of UpgradeServerID SHOULD be initialized to their default values before being used.

Table 11-10. Attributes of OTA Upgrade Cluster

Id	Name	Type	Range	Acc	Default	M/O
0x0000	*UpgradeServerID*	EUI64	-	R	0xffffffffffffffff	M
0x0001	*FileOffset*	uint32	*all*	R	0xffffffff	O
0x0002	*CurrentFileVersion*	uint32	*all*	R	0xffffffff	O
0x0003	*CurrentZigBeeStackVersion*	uint16	*all*	R	0xffff	O
0x0004	*DownloadedFileVersion*	uint32	*all*	R	0xffffffff	O
0x0005	*DownloadedZigBeeStackVersion*	uint16	*all*	R	0xffff	O
0x0006	*ImageUpgradeStatus*	enum8	*all*	R	0x00	M
0x0007	*Manufacturer ID*	uint16	*all*	R	-	O
0x0008	*Image Type ID*	uint16	*all*	R	-	O
0x0009	*MinimumBlockPeriod*	uint16	0x0000-0xfffe[127]	R	0[128]	O
0x000a	*Image Stamp*	uint32	*all*	R		O
0x000b	*UpgradeActivationPolicy*	enum8	0x00-0x01	R	0x00	O
0x000c	*UpgradeTimeout Policy*	enum8	0x00-0x01	R	0x00	O

[127] CCB 2398 *MinimumBlockPeriod* is milliseconds (not seconds)
[128] CCB 2398 *MinimumBlockPeriod* is milliseconds (not seconds)

11.10.1 UpgradeServerID Attribute

The attribute is used to store the IEEE address of the upgrade server resulted from the discovery of the upgrade server's identity. If the value is set to a non-zero value and corresponds to an IEEE address of a device that is no longer accessible, a device MAY choose to discover a new Upgrade Server depending on its own security policies.

The attribute is mandatory because it serves as a placeholder in a case where the client is programmed, during manufacturing time, its upgrade server ID. In addition, the attribute is used to identify the current upgrade server the client is using in a case where there are multiple upgrade servers in the network. The attribute is also helpful in a case when a client has temporarily lost connection to the network (for example, via a reset or a rejoin), it SHALL try to rediscover the upgrade server via network address discovery using the IEEE address stored in the attribute.

By default, the value is 0xffffffffffffffff, which is an invalid IEEE address. The attribute is a client-side attribute and stored on the client. Please refer to section 11.8 for a description of OTA server discovery.

11.10.2 *FileOffset* Attribute

The parameter indicates the current location in the OTA upgrade image. It is essentially the (start of the) address of the image data that is being transferred from the OTA server to the client. The attribute is optional on the client and is made available in a case where the server wants to track the upgrade process of a particular client.

11.10.3 CurrentFileVersion Attribute

The file version of the running firmware image on the device. The information is available for the server to query via ZCL read attribute command. The attribute is optional on the client.

11.10.4 CurrentZigBeeStackVersion Attribute

The ZigBee stack version of the running image on the device. The information is available for the server to query via ZCL read attribute command. The attribute is optional on the client. See 11.4.2.8 for values.

11.10.5 *DownloadedFileVersion* Attribute

The file version of the downloaded image on additional memory space on the device. The information is available for the server to query via ZCL read attribute command. The information is useful for the OTA upgrade management, so the server MAY ensure that each client has downloaded the correct file version before initiate the upgrade. The attribute is optional on the client.

11.10.6 DownloadedZigBeeStackVersion Attribute

The ZigBee stack version of the downloaded image on additional memory space on the device. The information is available for the server to query via ZCL read attribute command. The information is useful for the OTA upgrade management, so the server SHALL ensure that each client has downloaded the correct ZigBee stack version before initiate the upgrade. The attribute is optional on the client.

11.10.7 ImageUpgradeStatus Attribute

The upgrade status of the client device. The status indicates where the client device is at in terms of the download and upgrade process. The status helps to indicate whether the client has completed the download process and whether it is ready to upgrade to the new image. The status MAY be queried by the server via ZCL read attribute command. Hence, the server MAY not be able to reliably query the status of ZED client since the device MAY have its radio off.

Table 11-11. Image Upgrade Status Attribute Values

Image Upgrade Status Values	Description
0x00	Normal
0x01	Download in progress
0x02	Download complete
0x03	Waiting to upgrade
0x04	Count down
0x05	Wait for more
0x06	Waiting to Upgrade via External Event

Normal status typically means the device has not participated in any download process. Additionally, the client SHALL set its upgrade status back to Normal if the previous upgrade process was not successful.

Download in progress status is used from when the client device receives SUCCESS status in the Query Next Image Response command from the server prior to when the device receives all the image data it needs.

Download complete status indicates the client has received all data blocks required and it has already verified the OTA Upgrade Image signature (if applied) and has already written the image onto its additional memory space. The status will be modified as soon as the client receives Upgrade End Response command from the server.

Wait to upgrade status indicates that the client is told by the server to wait until another (upgrade) command is sent from the server to indicate the client to upgrade its image.

Count down status indicates that the server has notified the client to count down to when it SHALL upgrade its image.

Wait for more (upgrade) image indicates that the client is still waiting to receive more OTA upgrade image files from the server. This is true for a client device that is composed of multiple processors and each processor requires different image. The client SHALL be in this state until it has received all necessary OTA upgrade images, then it SHALL transition to Download complete state.

11.10.8 Manufacturer ID Attribute

This attribute SHALL reflect the ZigBee assigned value for the manufacturer of the device. See also section 11.4.2.5.

11.10.9 Image Type ID Attribute

This attribute SHALL indicate the image type identifier of the file that the client is currently downloading, or a file that has been completely downloaded but not upgraded to yet. The value of this attribute SHALL be 0xFFFF when the client is not downloading a file or is not waiting to apply an upgrade.

11.10.10 *MinimumBlockPeriod* Attribute

This attribute acts as a rate limiting feature for the server to slow down the client download and prevent saturating the network with block requests. The attribute lives on the client but can be changed during a download if rate limiting is supported by both devices.

This attribute SHALL reflect the minimum delay between Image Block Request commands generated by the client in milliseconds[129]. The value of this attribute SHALL be updated when the rate is changed by the server, but SHOULD reflect the client default when an upgrade is not in progress or a server does not support this feature.

11.10.11 Image Stamp Attribute

This attribute acts as a second verification to identify the image in the case that sometimes developers of the application have forgotten to increase the firmware version attribute. It is a 32 bit value and has a valid range from 0x00000000 to 0xFFFFFFFF. This attribute value must be consistent during the lifetime of the same image and also must be unique for each different build of the image. This attribute value SHOULD not be hardcoded or generated by any manual process. This attribute value SHOULD be generated by performing a hash or checksum on the entire image. There are two possible methods to generate this checksum. It can be generated dynamically during runtime of the application or it can be generated during compile time of the application.

11.10.12 UpgradeActivationPolicy Attribute

This attribute indicates what behavior the client device supports for activating a fully downloaded but not installed upgrade image. Table 11-12 below lists the enumerated values and the descriptions.

Table 11-12. UpgradeActivationPolicy Enumerations

Policy Enumeration Value	Short Name	Description
0x00	OTA Server Activation Allowed	This value indicates that the OTA server's command, to tell the device when to upgrade, will be applied by the client.
0x01	Out-of-band Activation Only	This value indicates that the activation of the image is done via out-of-band mechanisms. Attempts by the OTA server to tell the client to install the image will be rejected. Examples of an out-of-band mechanism to apply the image are: user prompt, or non-ZigBee protocol message.

Client devices with an *UpgradeActivationPolicy* value of 0x01 SHALL still send an *UpgradeEndRequest* command to the OTA Server at the completion of their download. In this case, clients SHALL NOT process an *UpgradeEndResponse* with a status of SUCCESS unless it has an UpgradeTime of 0xFFFFFFFF; upon receipt of an *UpgradeEndResponse* with a status of SUCCESS, but having an UpgradeTime field other than 0xFFFFFFFF, the client SHALL send back a Default Response with a status of NOT_AUTHORIZED.

In the absence of this optional attribute, the default value of 0x00 shall be assumed.

11.10.13 UpgradeTimeoutPolicy Attribute

This attribute indicates what behavior the client device supports for activating a fully downloaded image when the OTA server cannot be reached.

There may be circumstances when the OTA client is waiting on an explicit activation command and yet the activation command cannot be retrieved. This may be due to the fact that the OTA server is down, or, if *UpgradeActivationPolicy* is 0x01, the out-of-band communications mechanism is inaccessible.

[129] CCB 2398 *MinimumBlockPeriod* is milliseconds (not seconds)

In these circumstances the behavior of the device is dictated by the UpgradeTimeoutPolicy. After enough failed attempts to retrieve the activation command without any response, the OTA client's behavior SHALL be dictated by Table 11-13.

When the *UpgradeTimeoutPolicy* attribute is set to 0x00 and the *UpgradeActivationPolicy* is 0x00, section 11.16 defines the requirements on how often retries are performed and at what required intervals. If the *UpgradeTimeoutPolicy* attribute is set to 0x00 and the *UpgradeActivationPolicy* is 0x01, any retry mechanism and timeouts are manufacturer specific. Whilst there may be situations where the mechanisms defined in section 11.16 could be disabled when the *UpgradeActivationPolicy* is 0x00, the setting of the *UpgradeTimeoutPolicy* attribute to 0x01 in this case is currently reserved.

In the absence of this optional attribute, the default value of 0x00 shall be assumed.

Table 11-13. UpgradeTimeoutPolicy Enumerations

Policy Enumeration Value	Short Name	Description
0x00	Apply Upgrade After Timeout	After the specified time has elapsed and the number of required retry attempts has been made, the device SHALL apply a downloaded but not installed image.
0x01	Do not Apply Upgrade After Timeout	No amount of time or failed attempts to retrieve the activation command SHALL trigger the device to apply a downloaded but not installed image.

11.11 OTA Cluster Parameters

Below are defined parameters for OTA Upgrade cluster server. These values are considered as parameters and not attributes because their values tend to change often and are not static. Moreover, some of the parameters MAY have multiple values on the upgrade server at one instance. For example, for DataSize parameter, the value MAY be different for each OTA upgrade process. These parameters are included in commands sent from server to client. The parameters cannot be read or written via ZCL global commands.

Table 11-14. Parameters of OTA Upgrade Cluster

Name	Type	Range	Default	M/O
QueryJitter	uint8	0x01 – 0x64	0x32	M
DataSize	uint8	0x00 – 0xff	0xff	M
OTAImageData	Opaque	Varied	all 0xff's	M
CurrentTime	UTC[130]	*all*	0xffffffff	M
UpgradeTime or *RequestTime*	UTC[131]	*all*	0xffffffff	M

[130] CCB 2228 UTC time, not unsigned 32-bit integer
[131] CCB 2228 UTC time, not unsigned 32-bit integer

11.11.1 QueryJitter Parameter

The parameter is part of Image Notify Command sent by the upgrade server. The parameter indicates whether the client receiving Image Notify Command SHOULD send in Query Next Image Request command or not.

The server chooses the parameter value between 1 and 100 (inclusively) and includes it in the Image Notify Command. On receipt of the command, the client will examine other information (the manufacturer code and image type) to determine if they match its own values. If they do not, it SHALL discard the command and no further processing SHALL continue. If they do match, then it will determine whether or not it SHOULD query the upgrade server. It does this by randomly choosing a number between 1 and 100 and comparing it to the value of the QueryJitter parameter received. If it is less than or equal to the QueryJitter value from the server, it SHALL continue with the query process. If not, then it SHALL discard the command and no further processing SHALL continue.

By using the QueryJitter parameter, it prevents a single notification of a new OTA upgrade image from flooding the upgrade server with requests from clients.

11.11.2 *DataSize* Parameter

A value that indicates the length of the OTA image data included in the (Image Block Response) command payload sent from the server to client.

11.11.3 OTAImageData Parameter

This is a part of OTA upgrade image being sent over the air. The length of the data is dictated by the data size parameter. The server does not need to understand the meaning of the data, only the client does. The data MAY also be compressed or encrypted to increase efficiency or security.

The parameter is a series of octets and is used with the file offset value (defined in section 11.10.2) to indicate the location of the data and the data size value to indicate the length of the data.

11.11.4 CurrentTime and UpgradeTime/RequestTime Parameters

If CurrentTime and UpgradeTime are used in the command (ex. Upgrade End Response), the server uses the parameters to notify the client when to upgrade to the new image. If CurrentTime and RequestTime are used in the command (ex. Image Block Response), the server is notifying the client when to request for more upgrade data. The CurrentTime indicates the current time of the OTA server. The UpgradeTime indicates the time that the client SHALL upgrade to running new image. The RequestTime indicates when the client SHALL request for more data.

The value of the parameters and their interpretation MAY be different depending on whether the devices support ZCL Time cluster or not. If ZCL Time cluster is supported, the values of both parameters MAY indicate the UTC Time values that represent the Universal Time Coordinated (UTC) time. If the device does not support ZCL Time cluster, then it SHALL compute the offset time value from the difference between the two time parameters. The resulted offset time is in seconds. A device that does support the time cluster MAY use offset time instead of UTC Time when it sends messages that reference the time according to Table 11-15.

The table below shows how to interpret the time parameter values depending on whether Time cluster is supported on the device. The intention here is to be able to support a mixed network of nodes that MAY not all support Time cluster.

Table 11-15. Meaning of CurrentTime and UpgradeTime Parameters

CurrentTime Value	UpgradeTime or RequestTime Value	Description
0x00000000	Any	Device SHALL use UpgradeTime or RequestTime as an offset time from now.

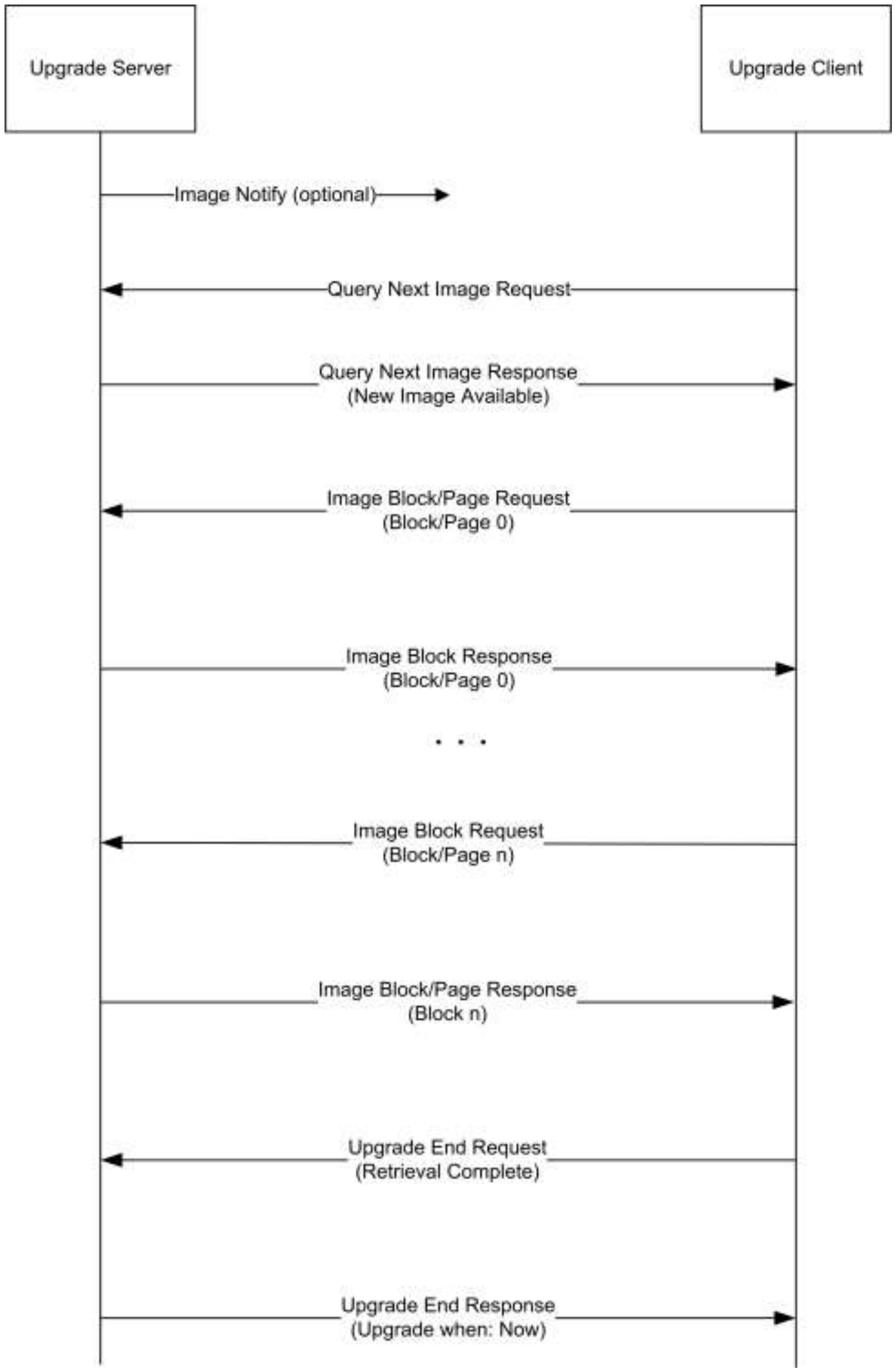
CurrentTime Value	**UpgradeTime or RequestTime Value**	**Description**
0x00000001 – 0xfffffffe	Any	Server supports Time cluster; client SHALL use UpgradeTime/RequestTime value as UTCTime if it also supports Time cluster or it SHALL compute the offset time if it does not.
Any	0xffffffff	The client SHOULD wait for a (upgrade) command from the server. Note that value of 0xffffffff SHOULD not be used for RequestTime.

For client devices with an *UpgradeActivationPolicy* value of 0x00. using a value of all 0xFF's for UpgradeTime to indicate a wait (for an Upgrade End Response command from the server) on ZED client devices is not recommended since an upgrade server SHOULD not be assumed to know the wake up cycle of the end device; hence it is not guaranteed that the end device will receive the upgrade command. If the wait value (0xffffffff) is used on a ZED client, the client SHOULD keep querying the server at a reasonable rate (not faster than once every 60 minutes) to see if it is time to upgrade. Client devices with an *UpgradeActivationPolicy* value of 0x01 will expect an UpgradeTime of all 0xFF's, but SHALL NOT keep querying the server.

Using value of all 0xFF's for RequestTime to indicate an indefinite wait time is not recommended. If the server does not know when it will have the image data ready, it SHALL use a reasonable wait time and when the client resends the image request, the server SHALL keep telling it to wait. There is no limit to how many times the server SHOULD the client to wait for the upgrade image. Using value of 0xffffffff SHALL cause the client to wait indefinitely and server MAY not have a way to tell the client to stop waiting especially for ZED client.

16664 11.12 OTA Upgrade Diagram

16665 **Figure 11-9. OTA Upgrade Message Diagram**


Upgrade Server
Upgrade Client
Image Notify (optional)
Query Next Image Request
Query Next Image Response (New Image Available)
Image Block/Page Request (Block/Page 0)
Image Block Response (Block/Page 0)
. . .
Image Block Request (Block/Page n)
Image Block/Page Response (Block n)
Upgrade End Request (Retrieval Complete)
Upgrade End Response (Upgrade when: Now)

16666

16667

Please refer to section 11.13 for the command description used in Figure 11-9.

11.13 Command Frames

OTA upgrade messages do not differ from typical ZigBee APS messages so the upgrade process SHOULD not interrupt the general network operation. All OTA Upgrade cluster commands SHALL be sent with APS retry option, hence, require APS acknowledgement; unless stated otherwise.

OTA Upgrade cluster commands, the frame control value SHALL follow the description below:

- Frame type is 0x01: commands are cluster specific (not a global command).
- Manufacturer specific is 0x00: commands are not manufacturer specific.
- Direction: SHALL be either 0x00 (client->server) or 0x01 (server->client) depending on the commands.
- Disable default response is 0x00 for all OTA request commands sent from client to server: default response command SHALL be sent when the server receives OTA Upgrade cluster request commands that it does not support or in case an error case happens. A detailed explanation of each error case along with its recommended action is described for each OTA cluster command.
- Disable default response is 0x01 for all OTA response commands (sent from server to client) and for broadcast/multicast Image Notify command: default response command is not sent when the client receives a valid OTA Upgrade cluster response commands or when it receives broadcast or multicast Image Notify command. However, if a client receives invalid OTA Upgrade cluster response command, a default response SHALL be sent. A detailed explanation of each error case along with its recommended action is described for each OTA cluster command.

11.13.1 OTA Cluster Command Identifiers

Command identifier values are listed in Table 11-16 below.

Table 11-16. OTA Upgrade Cluster Command Frames

Id	Name	Direction	Disable Default Response	M/O
0x00	*Image Notify*	Server -> Client(s) (0x01)	Set if sent as broadcast or multicast; Not Set if sent as unicast	O
0x01	*Query Next Image Request*	Client -> Server (0x00)	Not Set	M
0x02	*Query Next Image Response*	Server -> Client (0x01)	Set	M
0x03	*Image Block Request*	Client -> Server (0x00)	Not Set	M
0x04	*Image Page Request*	Client -> Server (0x00)	Not Set	O
0x05	*Image Block Response*	Server -> Client (0x01)	Set	M
0x06	*Upgrade End Request*	Client -> Server (0x00)	Not Set	M
0x07	*Upgrade End Response*	Server -> Client (0x01)	Set	M

Id	Name	Direction	Disable Default Response	M/O
0x08	*Query Device Specific File Request*	Client -> Server (0x00)	Not Set	O
0x09	*Query Device Specific File Response*	Server -> Client (0x01)	Set	O

11.13.2 OTA Cluster Status Codes

OTA Upgrade cluster uses ZCL defined status codes during the upgrade process. These status codes are included as values in status field in payload of OTA Upgrade cluster's response commands and in default response command. Some of the status codes are new and are still in the CCB process in order to be included in the ZCL specification.

Table 11-17. Status Code Defined and Used by OTA Upgrade Cluster

ZCL Status Code	Value	Description
SUCCESS	0x00	Success Operation
ABORT	0x95	Failed case when a client or a server decides to abort the upgrade process.
NOT_AUTHORIZED	0x7E	Server is not authorized to upgrade the client
INVALID_IMAGE	0x96	Invalid OTA upgrade image (ex. failed signature validation or signer information check or CRC check)
WAIT_FOR_DATA	0x97	Server does not have data block available yet
NO_IMAGE_AVAILABLE	0x98	No OTA upgrade image available for a particular client
MALFORMED_COMMAND	0x80	The command received is badly formatted. It usually means the command is missing certain fields or values included in the fields are invalid ex. invalid jitter value, invalid payload type value, invalid time value, invalid data size value, invalid image type value, invalid manufacturer code value and invalid file offset value
UNSUP_CLUSTER_COMMAND	0x81	Such command is not supported on the device
REQUIRE_MORE_IMAGE	0x99	The client still requires more OTA upgrade image files in order to successfully upgrade

11.13.3 Image Notify Command

The purpose of sending Image Notify command is so the server has a way to notify client devices of when the OTA upgrade images are available for them. It eliminates the need for ZR client devices having to check with the server periodically of when the new images are available. However, all client devices still need to send in Query Next Image Request command in order to officially start the OTA upgrade process.

11.13.3.1 Payload Format

Figure 11-10. Format of Image Notify Command Payload

Octets	**1**	**1**	**0/2**	**0/2**	**0/4**
Data Type	enum8	uint8	uint16	uint16	uint32
Field Name	Payload type	Query jitter	Manufacturer code	Image type	(new) File version

11.13.3.2 Payload Field Definitions

11.13.3.2.1 Image Notify Command Payload Type

Table 11-18. Image Notify Command Payload Type

Payload Type Values	**Description**
0x00	Query jitter
0x01	Query jitter and manufacturer code
0x02	Query jitter, manufacturer code, and image type
0x03	Query jitter, manufacturer code, image type, and new file version

11.13.3.2.2 Query Jitter

See section 11.11.1 for detailed description.

11.13.3.2.3 Manufacturer Code

Manufacturer code when included in the command SHOULD contain the specific value that indicates certain manufacturer. If the server intends for the command to be applied to all manufacturers, then the value SHOULD be omitted. See section 2 for detailed description.

11.13.3.2.4 Image Type

Image type when included in the command SHOULD contain the specific value that indicates certain file type. If the server intends for the command to be applied to all image type values then the wild card value (0xffff) SHOULD be used. See section 11.4.2.6 for detailed description.

11.13.3.2.5 (new) File Version

The value SHALL be the OTA upgrade file version that the server tries to upgrade client devices in the network to. If the server intends for the command to be applied to all file version values then the wild card value (0xffffffff) SHOULD be used. See section 11.10.3 for detailed description.

11.13.3.3 When Generated

For ZR client devices, the upgrade server MAY send out a unicast, broadcast, or multicast indicating it has the next upgrade image, via an Image Notify command. Since the command MAY not have APS security (if it is broadcast or multicast), it is considered purely informational and *not authoritative*. Even in the case of a unicast, ZR SHALL continue to perform the query process described in later section.

When the command is sent with payload type value of zero, it generally means the server wishes to notify all clients disregard of their manufacturers, image types or file versions. Query jitter is needed to protect the server from being flooded with clients' queries for next image.

The server MAY choose to send the Image Notify command to a more specific group of client devices by choosing higher payload type value. Only devices with matching information as the ones included in the Image Notify command will send back queries for next image.

However, payload type value of 0x03 has a slightly different effect. If the client device has all the information matching those included in the command including the new file version, the device SHALL then ignore the command. This indicates that the device has already gone through the upgrade process. This is to prevent the device from downloading the same image version multiple times. This is only true if the command is sent as broadcast/multicast.

Query jitter value indicates how the server wants to space out the responses from the client; generally as a result of sending the command as broadcast or multicast. The client will only respond back if it randomly picks a value that is equal or smaller than the query jitter value. When sending Image Notify command as broadcast or multicast, the Disable Default Response bit in ZCL header must be set (to 0x01) to avoid the client from sending any default response back to the upgrade server. This agrees with section 2.4.12.

If the command is sent as unicast, the payload type value MAY be zero and the Query jitter value MAY be the maximum value of 100 to signal the client to send in Query Next Image Request. The server MAY choose to use other payload type values besides zero when sending as unicast. However, since the server already knows the specific client (address) it wants to upgrade so other information is generally irrelevant.

The upgrade server MAY choose to send Image Notify command to avoid having ZR clients sending in Query Next Image Request to it periodically.

11.13.3.4 Effect on Receipt

On receipt of a unicast Image Notify command, the device SHALL always send a Query Next Image request back to the upgrade server.

On receipt of a broadcast or multicast Image Notify command, the device SHALL keep examining each field included in the payload with its own value. For each field, if the value matches its own, it SHALL proceed to examine the next field. If values in all three fields (naming manufacturer code, image type and new file version) match its own values, then it SHALL discard the command. The new file version in the payload SHALL be a match, it either matches the device's current running file version or matches the downloaded file version (on the additional memory space).

If manufacturer code or the image type values in the payload does not match the device's own value, it SHALL discard the command. For payload type value of 0x01, if manufacturer code matches the device's own value, the device SHALL proceed. For payload type value of 0x02, if both manufacturer code and image type match the device's own values, the device SHALL proceed. For payload type value of 0x03, if both manufacturer code and image type match the device's own values but the new file version is not a match, the device SHALL proceed. In this case, the (new) file version MAY be lower or higher than the device's file version to indicate a downgrade or an upgrade of the firmware respectively.

To proceed, the device SHALL randomly choose a number between 1 and 100 and compare it to the value of the QueryJitter value in the received message. If the generated value is less than or equal to the received value for QueryJitter, it SHALL query the upgrade server. If not, then it SHALL discard the message and no further processing SHALL continue.

By using the QueryJitter field, a server MAY limit the number of devices that will query it for a new OTA upgrade image, preventing a single notification of a new software image from flooding the upgrade server with requests.

In application standards that mandate APS encryption for OTA upgrade cluster messages, OTA messages sent as broadcast or multicast SHOULD be dropped by the receivers.

11.13.3.5 Handling Error Cases

The section describes all possible error cases that the client MAY detect upon reception invalid Image Notify command from the server, along with the action that SHALL be taken.

For invalid broadcast or multicast Image Notify command, for example, out-of-range query jitter value is used, or the reserved payload type value is used, or the command is badly formatted, the client SHALL ignore such command and no processing SHALL be done. In addition, the broadcast/multicast command SHALL have disable default response bit in the ZCL frame control set to 0x01.

The cases below describe how to handle invalid Image Notify command that is sent as unicast. Such command SHALL not have default response bit set.

11.13.3.5.1 Malformed Command

Scenarios for this error case include unicast Image Notify command with payload type of non-zero value, unicast Image Notify command with query jitter value that is not 100, broadcast Image Notify command with out of range query jitter value. In such scenario, the client SHOULD ignore the invalid message and SHALL send default response command with MALFORMED_COMMAND status to the server.

11.13.4 Query Next Image Request Command

11.13.4.1 Payload Format

Figure 11-11. Format of Query Next Image Request Command Payload

Octets	**1**	**2**	**2**	**4**	**0/2**
Data Type	map8[132]	uint16	uint16	uint32	uint16
Field Name	Field control	Manufacturer code	Image type	(Current) File version	Hardware version

11.13.4.2 Payload Field Definitions

11.13.4.2.1 Query Next Image Request Command Field Control

The field control indicates whether additional information such as device’s current running hardware version is included as part of the Query Next Image Request command.

Table 11-19. Query Next Image Request Field Control Bitmask

Bits	**Name**
0	Hardware Version Present

[132] CCB 2219 Field Control is a bitmap, not an unsigned integer

11.13.4.2.2 Manufacturer Code

The value SHALL be the device's assigned manufacturer code. Wild card value SHALL not be used in this case. See Chapter 2 for detailed description.

11.13.4.2.3 Image Type

The value SHALL be between 0x0000 - 0xffbf (manufacturer specific value range). See section 11.4.2.6 for detailed description. For other image type values, Query Device Specific File Request command SHOULD be used.

11.13.4.2.4 File Version (current)

The file version included in the payload represents the device's current running image version. Wild card value SHALL not be used in this case. See section 11.10.3 for more detailed description.

11.13.4.2.5 Hardware Version (optional)

The hardware version if included in the payload represents the device's current running hardware version. Wild card value SHALL not be used in this case. See section 11.4.2.13 for hardware version format description.

11.13.4.3 When Generated

Client devices SHALL send a Query Next Image Request command to the server to see if there is new OTA upgrade image available. ZR devices MAY send the command after receiving Image Notify command. ZED device SHALL periodically wake up and send the command to the upgrade server. Client devices query what the *next* image is, based on their own information.

11.13.4.4 Effect on Receipt

The server takes the client's information in the command and determines whether it has a suitable image for the particular client. The decision SHOULD be based on specific policy that is specific to the upgrade server and outside the scope of this document... However, a recommended default policy is for the server to send back a response that indicates the availability of an image that matches the manufacturer code, image type, and the highest available file version of that image on the server. However, the server MAY choose to upgrade or downgrade a clients' image, as its policy dictates. If client's hardware version is included in the command, the server SHALL examine the value against the minimum and maximum hardware versions included in the OTA file header.

How the server retrieves and stores the clients' file is also outside the scope of this document. The server MAY have a backend communication to retrieve the images or it MAY have database software to manage file storage.

11.13.4.5 Handling Error Cases

All error cases resulting from receiving Query Next Image Request command are handled by the corresponding Query Next Image Response command with the exception of the malformed request command described below that is handled by default response command. Please refer to section 11.13.5.3 for more information regarding how the Query Next Image response command is generated.

11.13.4.5.1 Malformed Command

Upon reception a badly formatted Query Next Image Request command, for example, the command is missing one of the payload fields; the server SHALL send default response command with MALFORMED_COMMAND status to the client and it SHALL not process the command further.

11.13.5 Query Next Image Response Command

11.13.5.1 Payload Format

Figure 11-12. Format of Query Next Image Response Command Payload

Octets	1	0/2	0/2	0/4	0/4
Data Type	enum8[133]	uint16	uint16	uint32	uint32
Field Name	Status	Manufacturer code	Image type	File version	Image size

11.13.5.2 Payload Field Definitions

11.13.5.2.1 Query Next Image Response Status

Only if the status is SUCCESS that other fields are included. For other (error) status values, only status field SHALL be present. See section 11.13.2 for a complete list and description of OTA Cluster status codes.

11.13.5.2.2 Manufacturer Code

The value SHALL be the one received by the server in the Query Next Image Request command. See Chapter 2 for detailed description.

11.13.5.2.3 Image Type

The value SHALL be the one received by the server in the Query Next Image Request command. See section 11.4.2.6 for detailed description.

11.13.5.2.4 File Version

The file version indicates the image version that the client is required to install. The version value MAY be lower than the current image version on the client if the server decides to perform a downgrade. The version value MAY not be the same as the client's current version. Reinstallation of the same software version is not supported. In general, the version value SHOULD be higher than the current image version on the client to indicate an upgrade. See section 11.4.2.7 for more description.

11.13.5.2.5 Image Size

The value represents the total size of the image (in bytes) including header and all sub-elements. See section 11.4.2.10 for more description.

[133] CCB 2220 Status is an enumeration, not an unsigned integer

11.13.5.3 When Generated

The upgrade server sends a Query Next Image Response with one of the following status: SUCCESS, NO_IMAGE_AVAILABLE or NOT_AUTHORIZED. When a SUCCESS status is sent, it is considered to be the explicit authorization to a device by the upgrade server that the device MAY upgrade to a specific software image.

A status of NO_IMAGE_AVAILABLE indicates that the server is authorized to upgrade the client but it currently does not have the (new) OTA upgrade image available for the client. For all clients (both ZR and ZED), they SHALL continue sending Query Next Image Requests to the server periodically until an image becomes available.

A status of NOT_AUTHORIZED indicates the server is not authorized to upgrade the client. In this case, the client MAY perform discovery again to find another upgrade server. The client MAY implement an intelligence to avoid querying the same unauthorized server.

11.13.5.4 Effect on Receipt

A status of SUCCESS in the Query Next Image response indicates to the client that the server has a new OTA upgrade image. If the file version contained in the Query Next Image Response is the same as the CurrentFileVersion attribute (the current running version of software) or the *DownloadedFileVersion attribute for the specified Image Type*[134], then the message SHOULD[135] be discarded and no further processing SHOULD[136] be done. Reinstallation of the same software version is not supported. Otherwise the client MAY[137] begin requesting blocks of the image using the Image Block Request command. A ZED client MAY choose to change its wake cycle to retrieve the image more quickly.

11.13.5.5 Handling Error Cases

The Query Next Image Response command SHALL have the disable default response bit set. Hence, if the command is received successfully, no default response command SHALL be generated. However, the default response SHALL be generated to indicate the error cases below.

11.13.5.5.1 Malformed Command

Upon reception a badly formatted Query Next Image Response command, for example, the command is missing one of the payload field, other payload fields are included when the status field is not SUCCESS, the image type value included in the command does not match that of the device or the manufacturer code included in the command does not match that of the device; the client SHOULD ignore the message and SHALL send default response command with MALFORMED_COMMAND status to the server.

[134] CCB 2464 reject same image
[135] CCB 2464 reject same image
[136] CCB 2464 reject same image
[137] CCB 2342 allow a client to reject a downgraded image

11.13.6 Image Block Request Command

11.13.6.1 Payload Format

Figure 11-13. Format of Image Block Request Command Payload

Octets	**1**	**2**	**2**	**4**	**4**	**1**	**0/8**	**0/2**
Data Type	map8[138]	uint16	uint16	uint32	uint32	uint8	EUI64	uint16
Field Name	Field control	Manufacturer code	Image type	File version	File offset	Maximum data size	Request node address	MinimumBlockPeriod

11.13.6.2 Payload Field Definitions

11.13.6.2.1 Image Block Request Command Field Control

Field control value is used to indicate additional optional fields that MAY be included in the payload of Image Block Request command. Currently, the device is only required to support field control value of 0x00; support for other field control value is optional. A device SHALL process commands issued with unimplemented/unrecognized field control bits set. Devices SHALL correctly process messages containing fields indicated by unrecognized/unimplemented field control bits.

Field control value 0x00 (bit 0 not set) indicates that the client is requesting a generic OTA upgrade file; hence, there is no need to include additional fields. The value of Image Type included in this case SHALL be manufacturer specific.

Field control value of 0x01 (bit 0 set) means that the client's IEEE address is included in the payload. This indicates that the client is requesting a device specific file such as security credential, log or configuration; hence, the need to include the device's IEEE address in the image request command. The value of Image type included in this case SHALL be one of the reserved values that are assigned to each specific file type.

Table 11-20. Image Block Request Field Control Bitmask

Bits	**Name**
0	Request node's IEEE address Present
1	MinimumBlockPeriod present

11.13.6.2.2 Manufacturer Code

The value SHALL be that of the client device assigned to each manufacturer by ZigBee. See Chapter 2 for detailed description.

[138] CCB 2221 Field Control is a bitmap, not an unsigned integer

11.13.6.2.3 Image Type

The value SHALL be between 0x0000 - 0xffbf (manufacturer specific value range). See section 11.4.2.6 for detailed description.

11.13.6.2.4 File Version

The file version included in the payload represents the OTA upgrade image file version that is being requested. See section 11.4.2.7 for more detailed description.

11.13.6.2.5 File Offset

The value indicates number of bytes of data offset from the beginning of the file. It essentially points to the location in the OTA upgrade image file that the client is requesting the data from. The value reflects the amount of (OTA upgrade image file) data (in bytes) that the client has received so far.

See section 11.10.2 for more description.

11.13.6.2.6 Maximum Data Size

The value indicates the largest possible length of data (in bytes) that the client can receive at once. The server SHALL respect the value and not send the data that is larger than the maximum data size. The server MAY send the data that is smaller than the maximum data size value, for example, to account for source routing payload overhead if the client is multiple hops away. By having the client send both file offset and maximum data size in every command, it eliminates the burden on the server for having to remember the information for each client.

11.13.6.2.7 Request Node Address (optional)

This is the IEEE address of the client device sending the Image Block Request command.

11.13.6.2.8 MinimumBlockPeriod (optional)

This is the current value of the *MinimumBlockPeriod* attribute of the device that is making the request as set by the server. If the device supports the attribute, then it SHALL include this field in the request. The value is in milliseconds[139].

This attribute does not necessarily reflect the actual delay applied by the client between Image Block Requests, only the value set by the server on the client.

11.13.6.3 When Generated

The client device requests the image data at its leisure by sending Image Block Request command to the upgrade server. The client knows the total number of request commands it needs to send from the image size value received in Query Next Image Response command.

The client repeats Image Block Requests until it has successfully obtained all data. Manufacturer code, image type and file version are included in all further queries regarding that image. The information eliminates the need for the server to remember which OTA Upgrade Image is being used for each download process.

If the client supports the *MinimumBlockPeriod* attribute it SHALL include the value of the attribute as the MinimumBlockPeriod field of the Image Block Request message. The client SHALL ensure that it delays at least MinimumBlockPeriod after the previous Image Block Request was sent before sending the next Image Block Request message. A client MAY delay its next Image Block Requests longer than its MinimumBlockPeriod attribute.

[139] CCB 2398 *MinimumBlockPeriod* is milliseconds (not seconds)

11.13.6.4 Effect on Receipt

The server uses the manufacturer code, image type, and file version to uniquely identify the OTA upgrade image request by the client. It uses the file offset to determine the location of the requested data within the OTA upgrade image. If the server supports rate-limited transfers it SHALL check the MinimumBlockPeriod[140] field and compare it to the desired rate for the client. If the server receives an Image Block Request with a field control mask of 0x02, (i.e., MinimumBlockPeriod present) and the server does not support rate-limited transfers the server SHALL ignore the MinimumBlockPeriod value and process the command.

11.13.6.5 Handling Error Cases

In most cases, the server sends Image Block Response command in response to the client's Image Block Request command. However, with the exception of a few error cases described below that the server SHALL send default response command as a response.

11.13.6.5.1 Malformed Command

Upon reception a badly formatted Image Block Request command, for example, the command is missing one of the payload field or the file offset value requested by the client is invalid, for example, the value is larger than the total image size; the server SHOULD ignore the message and it SHALL send default response command with MALFORMED_COMMAND status to the client.

11.13.6.5.2 No Image Available

If either manufacturer code or image type or file version information in the request command is invalid or the OTA upgrade file for the client for some reason has disappeared which result in the server no longer able to retrieve the file, it SHALL send default response command with NO_IMAGE_AVAILABLE status to the client. After three attempts, if the client keeps getting the default response with the same status, it SHOULD go back to sending Query Next Image Request periodically or waiting for next Image Notify command.

11.13.6.5.3 Command Not Supported

If the client sends image request command with field control value of 0x01 that indicates device specific file request and if the server does not support such request, it SHALL send default response with UNSUP_CLUSTER_COMMAND status. Upon reception of such response, the client SHOULD terminate the attempt to request the device specific file and it MAY try to query different server.

11.13.7 Image Page Request Command

11.13.7.1 Payload Format

Figure 11-14. Image Page Request Command Payload

Octets	**1**	**2**	**2**	**4**	**4**	**1**	**2**	**2**	**0/8**
Data Type	map8[141]	uint16	uint16	uint32	uint32	uint8	uint16	uint16	EUI64

[140] CCB 2307 align attribute names with field names
[141] CCB 2219 Field Control is a bitmap, not an unsigned integer

Octets	1	2	2	4	4	1	2	2	0/8
Field Name	Field control	Manufacturer code	Image type	File version	File offset	Maximum data size	Page size	Response Spacing	Request node address

11.13.7.2 Payload Field Definitions

11.13.7.2.1 Image Page Request Command Field Control

Field control value is used to indicate additional optional fields that MAY be included in the payload of Image Page Request command. Currently, the device is only required to support field control value of 0x00; support for other field control value is optional.

Field control value 0x00 indicates that the client is requesting a generic OTA upgrade file; hence, there is no need to include additional fields. The value of Image Type included in this case SHALL be manufacturer specific.

Field control value of 0x01 means that the client's IEEE address is included in the payload. This indicates that the client is requesting a device specific file such as security credential, log or configuration; hence, the need to include the device's IEEE address in the image request command. The value of Image type included in this case SHALL be one of the reserved values that are assigned to each specific file type.

Table 11-21. Image Page Request Field Control Bitmask

Bits	Name
0	Request node's IEEE address Present

11.13.7.2.2 Manufacturer Code

The value SHALL be that of the client device assigned to each manufacturer by ZigBee. See Chapter 2 for detailed description.

11.13.7.2.3 Image Type

The value SHALL be between 0x0000 **-** 0xffbf (manufacturer specific value range). See section 11.4.2.6 for detailed description.

11.13.7.2.4 File Version

The file version included in the payload represents the OTA upgrade image file version that is being requested. See section 11.4.2.7 for more detailed description.

11.13.7.2.5 File Offset

The value indicates number of bytes of data offset from the beginning of the file. It essentially points to the location in the OTA upgrade image file that the client is requesting the data from. The value reflects the amount of (OTA upgrade image file) data (in bytes) that the client has received so far.

See section 11.10.2 for more description.

11.13.7.2.6 Maximum Data Size

The value indicates the largest possible length of data (in bytes) that the client can receive at once. The server SHALL respect the value and not send the data that is larger than the maximum data size. The server MAY send the data that is smaller than the maximum data size value, for example, to account for source routing payload overhead if the client is multiple hops away. By having the client send both file offset and maximum data size in every command, it eliminates the burden on the server for having to remember the information for each client.

11.13.7.2.7 Page Size

The value indicates the number of bytes to be sent by the server before the client sends another Image Page Request command. In general, page size value SHALL be larger than the maximum data size value.

11.13.7.2.8 Response Spacing

The value indicates how fast the server SHALL send the data (via Image Block Response command) to the client. The value is determined by the client. The server SHALL wait at the minimum the (response) spacing value before sending more data to the client. The value is in milliseconds.

11.13.7.2.9 Request Node Address (optional)

This is the IEEE address of the client device sending the Image Block Request command.

11.13.7.3 When Generated

The support for the command is optional. The client device MAY choose to request OTA upgrade data in one page size at a time from upgrade server. Using Image Page Request reduces the numbers of requests sent from the client to the upgrade server, compared to using Image Block Request command. In order to conserve battery life a device MAY use the Image Page Request command. Using the Image Page Request command eliminates the need for the client device to send Image Block Request command for every data block it needs; possibly saving the transmission of hundreds or thousands of messages depending on the image size.

The client keeps track of how much data it has received by keeping a cumulative count of each data size it has received in each Image Block Response. Once the count has reach the value of the page size requested, it SHALL repeat Image Page Requests until it has successfully obtained all pages. Note that the client MAY choose to switch between using Image Block Request and Image Page Request during the upgrade process. For example, if the client does not receive all data requested in one Image Page Request, the client MAY choose to request the missing block of data using Image Block Request command, instead of requesting the whole page again.

Since a single Image Page Request MAY result in multiple Image Block Response commands sent from the server, the client, especially ZED client, SHOULD make its best effort to ensure that all responses are received. A ZED client MAY select a small value for the response spacing and stay awake to receive all data blocks. Or it MAY choose a larger value and sleeps between receiving each data block.

Manufacturer code, image type and file version are included in all further queries regarding that image. The information eliminates the need for the server to remember which OTA Upgrade Image is being used for each download process.

11.13.7.4 Effect on Receipt

The server uses the file offset value to determine the location of the requested data within the OTA upgrade image. The server MAY respond to a single Image Page Request command with possibly multiple Image Block Response commands; depending on the value of page size. Each Image Block Response command sent as a result of Image Page Request command SHALL have increasing ZCL sequence number. Note that the sequence number MAY not be sequential (for example, if the server is also upgrading another client simultaneously); additionally ZCL sequence numbers are only 8-bit and MAY wrap.

In response to the Image Page Request, the server SHALL send Image Block Response commands with no APS retry to disable APS acknowledgement. The intention is to minimize the number of packets sent by the client in order to optimize the energy saving. APS acknowledgement is still used for Image Block Response sent in response to Image Block Request command.

Image Block Response message (in response to Image Page Request) only relies on network level retry. This MAY not be as reliable over multiple hops communication, however, the benefit of using Image Page Request is to save energy on the ZED client and using APS ack with the packet undermines that effort. ZED client needs to make the decision which request it uses. Image Page Request MAY speed up the upgrade process; the client transmits fewer packets, hence, less energy use but it MAY be less reliable. On the other hand, Image block request MAY slow down the upgrade process; the client is required to transmit more packets but it is also more predictable and reliable; it also allows the upgrade process to proceed at the client's pace.

11.13.7.5 Handling Error Cases

In most cases, the server sends Image Block Response command in response to the client’s Image Page Request command. However, with the exception of a few error cases described below that the server SHALL send default response command as a response.

11.13.7.5.1 Malformed Command

Upon reception a badly formatted Image Page Request command, for example, the command is missing one of the payload fields or the file offset value requested by the client is invalid. The server SHOULD ignore the message and it SHALL send default response command with MALFORMED_COMMAND status to the client.

11.13.7.5.2 No Image Available

If either manufacturer code or image type or file version information in the request command is invalid or the OTA upgrade file for the client for some reason has disappeared which result in the server no longer able to retrieve the file, it SHALL send default response command with NO_IMAGE_AVAILABLE status to the client. After three attempts, if the client keeps getting the default response with the same status, it SHOULD go back to sending Query Next Image Request periodically or waiting for next Image Notify command.

11.13.7.5.3 Command Not Supported

If the client sends Image Page Request command with field control value of 0x00 to request OTA upgrade image and the server does not support Image Page Request command, it SHALL send default response with UNSUP_CLUSTER_COMMAND status. Upon reception of such response, the client SHALL switch to using Image Block Request command instead to request OTA image data.

If the client sends image request command with field control value of 0x01 that indicates device specific file request and if the server does not support such request, it SHALL send default response with UNSUP_CLUSTER_COMMAND status. Upon reception of such response, the client SHOULD terminate the attempt to request the device specific file and it MAY try to query different server.

11.13.8 Image Block Response Command

11.13.8.1 Payload Format

Figure 11-15. Image Block Response Command Payload with SUCCESS status

Octets	1	2	2	4	4	1	Variable

Data Type	enum8[142]	uint16	uint16	uint32	uint32	uint8	Opaque
Field Name	Success status	Manufacturer code	Image type	File version	File offset	Data size	Image data

Figure 11-16. Image Block Response Command Payload with WAIT_FOR_DATA status

Octets	**1**	**4**	**4**	**2**
Data Type	enum8[143]	uint32	uint32	uint16
Field Name	Wait for data Status	Current time	Request time	MinimumBlockPeriod

Figure 11-17. Image Block Response Command Payload with ABORT status

Octets	**1**
Data Type	enum8[144]
Field Name	Abort Status

11.13.8.2 Payload Field Definitions

11.13.8.2.1 Image Block Response Status

The status in the Image Block Response command MAY be SUCCESS, ABORT or WAIT_FOR_DATA. If the status is ABORT then only the status field SHALL be included in the message, all other fields SHALL be omitted.

See section 11.13.2 for a complete list and description of OTA Cluster status codes.

11.13.8.2.2 Manufacturer Code

The value SHALL be the same as the one included in Image Block/Page Request command. See Chapter 2 for detailed description.

[142] CCB 2223 Status is an enumeration, not an unsigned integer
[143] CCB 2223 Status is an enumeration, not an unsigned integer
[144] CCB 2227 Status is an enumeration, not an unsigned integer

11.13.8.2.3 Image Type

The value SHALL be the same as the one included in Image Block/Page Request command. See section 11.4.2.6 for detailed description.

11.13.8.2.4 File Version

The file version indicates the image version that the client is required to install. The version value MAY be lower than the current image version on the client if the server decides to perform a downgrade. The version value MAY not be the same as the client's current version. Reinstallation of the same software version is not supported. However, in general, the version value SHOULD be higher than the current image version on the client to indicate an upgrade. See section 11.4.2.7 for more description.

11.13.8.2.5 File Offset

The value represents the location of the data requested by the client. For most cases, the file offset value included in the (Image Block) response SHOULD be the same as the value requested by the client. For (unsolicited) Image Block responses generated as a result of Image Page Request, the file offset value SHALL be incremented to indicate the next data location.

11.13.8.2.6 Data Size

The value indicates the length of the image data (in bytes) that is being included in the command. The value MAY be equal or smaller than the maximum data size value requested by the client. See section 11.11.2 for more description.

11.13.8.2.7 Image Data

The actual OTA upgrade image data with the length equals to data size value. See section 11.11.3 for more description.

11.13.8.2.8 Current Time and Request Time

If status is WAIT_FOR_DATA, the payload then includes the server's current time and the request time that the client SHALL retry the request command. The client SHALL wait at least the request time value before trying again. In case of sleepy device, it MAY choose to wait longer than the specified time in order to not disrupt its sleeping cycle. If the current time value is zero that means the server does not support UTC time and the client SHALL treat the request time value as offset time. If neither time value is zero, and the client supports UTC time, it SHALL treat the request time value as UTC time. If the client does not support UTC time, it SHALL calculate the offset time from the difference between the two time values. The offset indicates the minimum amount of time to wait in seconds. The UTC time indicates the actual time moment that needs to pass before the client SHOULD try again. See section 11.15.4 for more description.

11.13.8.2.9 MinimumBlockPeriod

This value is only included if the status is WAIT_FOR_DATA and the server supports rate limiting. This is the minimum delay that the server wants the client to wait between subsequent block requests. The client SHALL update its *MinimumBlockPeriod* attribute to this value. The MinimumBlockPeriod field value SHALL be observed in all future Image Block Request messages for the duration of the firmware image download, or until updated by the server.

If the server does not support rate limiting or does not wish to slow the client's download, the field SHALL be set to 0.

The client SHALL check the existence of this field by looking at the length of the message. If the field does not exist, then the field SHALL have the value of zero.[145]

[145] CCB 2296 this field was added as mandatory and older servers may not include it

See 11.10.10 for more description of the valid ranges and use of this attribute.

See section 11.15.3 for more description on how the rate limiting feature works.

11.13.8.3 When Generated

Upon receipt of an Image Block Request command the server SHALL generate an Image Block Response. If the server is able to retrieve the data for the client and does not wish to change the image download rate, it will respond with a status of SUCCESS and it will include all the fields in the payload. The use of file offset allows the server to send packets with variable data size during the upgrade process. This allows the server to support a case when the network topology of a client MAY change during the upgrade process, for example, mobile client MAY move around during the upgrade process. If the client has moved a few hops away, the data size SHALL be smaller. Moreover, using file offset eliminates the need for data padding since each Image Block Response command MAY contain different data size. A simple server implementation MAY choose to only support largest possible data size for the worst-case scenario in order to avoid supporting sending packets with variable data size.

The server SHALL respect the maximum data size value requested by the client and SHALL not send the data with length greater than that value. The server MAY send the data with length smaller than the value depending on the network topology of the client. For example, the client MAY be able to receive 100 bytes of data at once so it sends the request with 100 as maximum data size. But after considering all the security headers (perhaps from both APS and network levels) and source routing overhead (for example, the client is five hops away), the largest possible data size that the server can send to the client SHALL be smaller than 100 bytes.

If the server simply wants to cancel the download process, it SHALL respond with ABORT status. An example is while upgrading the client the server MAY receive newer image for that client. It MAY then choose to abort the current process so that the client MAY reinitiate a new upgrade process for the newer image.

If the server does not have the image block available for the client yet or it wants to slow down (pause or rate-limit) the download process, it SHALL send the response back with status WAIT_FOR_DATA and with RequestTime value that the client SHALL wait before resending the request. This is a one-time (temporary) delay of the download for the client.

If the Image Block Request message contains the MinimumBlockPeriod field and the server wishes to slow the client's rate of sending Image Block requests, then the server SHALL send an Image Block Response with status WAIT_FOR_DATA. In this case the RequestTime and CurrentTime in the message SHALL be set so that their delta is zero, and the MinimumBlockPeriod field SHALL be set to the minimum delay that server wants the client to add between all subsequent Image Block Requests.

11.13.8.4 Effect on Receipt

When the client receives the Image Block Response it SHALL examine the status field. If the value is SUCCESS, it SHALL write the image data to its additional memory space. The client then SHALL continue to send Image Block Request commands with incrementing block numbers to request the remaining blocks of the OTA upgrade image. If the client has received the final block of the image, it SHALL generate an Upgrade End request command. In case of the client using Image Page Request, after receiving an Image Block Response, the client SHALL wait for response spacing time before expecting another Image Block Response from the server. A ZED client MAY go to sleep in between receiving Image Block Responses in order to save the energy.

If the client receives a response with ABORT status, it SHALL abort the upgrade process. It MAY retry the entire upgrade operation at a later point in time.

Upon receipt of WAIT_FOR_DATA status, the client SHALL wait at a minimum for the specified RequestTime and try to retrieve the image data again by resending Image Block Request or Image Page Request command with the same file offset value. If the CurrentTime and RequestTime are the same value and the client supports the *MinimumBlockPeriod* attribute, then it SHALL examine if the message contains the MinimumBlockPeriod[146] field in the Image Block Response. If the field is present and has a value is different than its current attribute value, it SHALL update its local attribute. Prior to sending its next Image Block Request message it SHALL add a minimum delay equal to the new value of its *MinimumBlockPeriod*[147] attribute.

If the delta between the CurrentTime and RequestTime is zero and the MinimumBlockPeriod field is not present or is zero, the client MAY immediately send an Image Block Request command.

11.13.8.5 Handling Error Cases

If Image Block Response command is received successfully by the client, no default response will be generated if the disable default response bit is set in the ZCL header. However, a few error cases described below MAY cause the client to send default response to the server with an error code.

11.13.8.5.1 Malformed Command

Upon reception a badly formatted Image Block Response command, for example, the command is missing one of the payload field, the payload fields do not correspond to the status field, the request time value returned by the server is invalid, for example, the value is less than the client's current time or the value is less than the server's own current time, the data size value returned by the server is invalid, for example, the value is greater than the maximum data size specified by the client, or the value does not match the number of bytes of data actually included in the payload, or the value, when combined with file offset, is greater than the total image size or the file offset value returned by the server is invalid. The client SHOULD ignore the command and SHALL send default response command with MALFORMED_COMMAND status to the server.

11.13.9 Upgrade End Request Command

11.13.9.1 Payload Format

Figure 11-18. Format of Upgrade End Request Command Payload

Octets	**1**	**2**	**2**	**4**
Data Type	enum8[148]	uint16	uint16	uint32
Field Name	Status	Manufacturer code	Image type	File version

11.13.9.2 Payload Field Definitions

11.13.9.2.1 Upgrade End Request Command Status

The status value of the Upgrade End Request command SHALL be SUCCESS, INVALID_IMAGE, REQUIRE_MORE_IMAGE, or ABORT. See section 11.13.2 for more description.

[146] CCB 2307 align attribute names with field names
[147] CCB 2307 align attribute names with field names
[148] CCB 2224 Status is an enumeration, not an unsigned integer

11.13.9.2.2 Manufacturer Code

The value SHALL be that of the client device assigned to each manufacturer by ZigBee. See Chapter 2 for detailed description.

11.13.9.2.3 Image Type

The value SHALL be between 0x0000 - 0xffbf (manufacturer specific value range). See section 11.4.2.6 for detailed description.

11.13.9.2.4 File Version

The file version included in the payload represents the newly downloaded OTA upgrade image file version. See section 11.4.2.7 for more detailed description.

11.13.9.3 When Generated

Upon reception all the image data, the client SHOULD verify the image to ensure its integrity and validity. If the device requires signed images it SHALL examine the image and verify the signature as described in section 11.7.1. Clients MAY perform additional manufacturer specific integrity checks to validate the image, for example, CRC check on the actual file data.

If the image fails any integrity checks, the client SHALL send an Upgrade End Request command to the upgrade server with a status of INVALID_IMAGE. In this case, the client MAY reinitiate the upgrade process in order to obtain a valid OTA upgrade image. The client SHALL not upgrade to the bad image and SHALL discard the downloaded image data.

If the image passes all integrity checks and the client does not require additional OTA upgrade image file, it SHALL send back an Upgrade End Request with a status of SUCCESS. However, if the client requires multiple OTA upgrade image files before performing an upgrade, it SHALL send an Upgrade End Request command with status REQUIRE_MORE_IMAGE. This SHALL indicate to the server that it cannot yet upgrade the image it received.

If the client decides to cancel the download process for any other reasons, it has the option of sending Upgrade End Request with status of ABORT at any time during the download process. The client SHALL then try to reinitiate the download process again at a later time.

When a client finishes downloading a device specific file, it SHALL send Upgrade End Request command with status of SUCCESS to the server to indicate the end of the upgrade process.

11.13.9.4 Effect on Receipt

For manufacturer specific image type file download, upon receipt of a SUCCESS Upgrade End Request command the upgrade server SHALL reply with the Upgrade End Response indicating when the client SHALL upgrade to the newly retrieved image. For other status value received such as INVALID_IMAGE, REQUIRE_MORE_IMAGE, or ABORT, the upgrade server SHALL not send Upgrade End Response command but it SHALL send default response command with status of success and it SHALL wait for the client to reinitiate the upgrade process.

The server MAY utilize the Upgrade End Request command as a means to know when devices are done downloading a particular image. This helps the server manage the images and remove those that are no longer needed. However, the upgrade server SHOULD not rely on receiving the command and MAY impose upper limits on how long it will store a particular OTA upgrade image. The specific implementation of this is outside the scope of this document.

11.13.9.5 Handling Error Cases

Upgrade End Request command does not have disable default response bit set. Hence, in a case where the Upgrade End Request command has been received and the server does not send Upgrade End Response command in response, a default response command SHALL be sent with SUCCESS status. If the Upgrade End Request command has not been received, default response command with error status SHALL be sent as described below.

11.13.9.5.1 Malformed Command

Upon reception a badly formatted Upgrade End Request command, for example, the command is missing one of the payload fields. The server SHALL send default response command with MALFORMED_COMMAND status to the client.

11.13.9.6 Upgrade End Response Command

11.13.9.6.1 Payload Format

Figure 11-19. Format of Upgrade End Response Command Payload

Octets	**2**	**2**	**4**	**4**	**4**
Data Type	uint16	uint16	uint32	UTC[149]	UTC[150]
Field Name	Manufacturer code	Image type	File version	Current time	Upgrade time

11.13.9.6.2 Payload Field Definitions

The ability to send the command with wild card values for manufacturer code, image type and file version is useful in this case because it eliminates the need for the server having to send the command multiple times for each manufacturer as well as having to keep track of all devices' manufacturers in the network.

11.13.9.6.3 Manufacturer Code

Manufacturer code MAY be sent using wildcard value of 0xffff in order to apply the command to all devices disregard of their manufacturers. See Chapter 2 for detailed description.

11.13.9.6.4 Image Type

Image type MAY be sent using wildcard value of 0xffff in order to apply the command to all devices disregard of their manufacturers. See section 11.4.2.6 for detailed description.

[149] CCB 2226 UTC time, not unsigned 32-bit integer
[150] CCB 2225 UTC time, not unsigned 32-bit integer

11.13.9.6.5 File Version

The file version included in the payload represents the newly downloaded OTA upgrade image file version. The value SHALL match that included in the request. Alternatively, file version MAY be sent using wildcard value of 0xffffffff in order to apply the command to all devices disregard of their manufacturers. See section 11.4.2.7 for more detailed description.

Current Time and Upgrade Time

Current time and Upgrade time values are used by the client device to determine when to upgrade its running firmware image(s) with the newly downloaded one(s). See section 11.15.4 for more description.

11.13.9.7 When Generated

When an upgrade server receives an Upgrade End Request command with a status of INVALID_IMAGE, REQUIRE_MORE_IMAGE, or ABORT, no additional processing SHALL be done in its part. If the upgrade server receives an Upgrade End Request command with a status of SUCCESS, it SHALL generate an Upgrade End Response with the manufacturer code and image type received in the Upgrade End Request along with the times indicating when the device SHOULD upgrade to the new image.

The server MAY send an unsolicited Upgrade End Response command to the client. This MAY be used for example if the server wants to synchronize the upgrade on multiple clients simultaneously. For client devices, the upgrade server MAY unicast or broadcast Upgrade End Response command indicating a single client device or multiple client devices SHALL switch to using their new images. The command MAY not be reliably received by sleepy devices if it is sent unsolicited.

For device specific file download, the client SHOULD not always expect the server to respond back with Upgrade End Response command. For example, in a case of a client has just finished retrieving a log file from the server, the server MAY not need to send Upgrade End Response command. However, if the client has just retrieved a security credential or a configuration file, the server MAY send Upgrade End Response command to notify the client of when to apply the file. The decision of whether Upgrade End Response command SHOULD be sent for device specific file download is manufacturer specific.

11.13.9.8 Effect on Receipt

The client SHALL examine the manufacturer code, image type and file version to verify that they match its own. If the received values do not match its own values or they are not wild card values, then it SHALL discard the command and no further processing SHALL continue. If all values match, the client SHALL examine the time values to determine the upgrade time. For more information on determining the time, please refer to section 11.15.4.

11.13.9.9 Handling Error Cases

If Upgrade End Response command is received successfully by the client or if it is sent as broadcast or multicast, no default response will be generated. However, a few error cases described below MAY cause the client to send default response to the server.

11.13.9.9.1 Malformed Command

Upon reception a badly formatted Upgrade End Response command, for example, the command is missing one of the payload field or the request time value returned by the server is invalid, for example, the value is less than the client's current time or the value is less than the server's own current time. The client SHOULD ignore the command and SHALL send default response command with MALFORMED_COMMAND status to the server.

11.13.10 Query Device Specific File Request Command

11.13.10.1 Payload Format

Figure 11-20. Format of Query Device Specific File Request Command Payload

Octets	**8**	**2**	**2**	**4**	**2**
Data Type	EUI64	uint16	uint16	uint32	uint16
Field Name	Request node address	Manufacturer code	Image type	File version	(Current) ZigBee stack version

11.13.10.2 Payload Field Definitions

11.13.10.2.1 Request Node Address

This is the IEEE address of the client device sending the request command. This indicates that the client is requesting a device specific file such as security credential, log or configuration; hence, the need to include the device's IEEE address in the image request command.

11.13.10.2.2 Manufacturer Code

The value SHALL be that of the client device assigned to each manufacturer by ZigBee. See Chapter 2 for detailed description.

11.13.10.2.3 Image Type

The value of image type included in this case SHALL be one of the reserved values that are assigned to each specific file type. The value SHOULD be between 0xffc0 – 0xfffe, however, only 0xffc0 – 0xffc2 is being used currently. See section 11.4.2.6 for detailed description.

11.13.10.2.4 File Version

The value indicates the version of the device specific file being requested. See section 11.4.2.7 for more detailed description.

11.13.10.2.5 (current) ZigBee Stack Version

The value MAY represent the current running ZigBee stack version on the device or the ZigBee stack version of the OTA upgrade image being stored in additional memory space. The decision of which value to include depends on which device specific file being requested. For example, if the client is requesting a new security credential file in order to be able to run the newly downloaded image (ex. SE 2.0), then it SHOULD include the ZigBee stack version value of the new image.

11.13.10.3 When Generated

Client devices SHALL send a Query Device Specific File Request command to the server to request for a file that is specific and unique to it. Such file could contain non-firmware data such as security credential (needed for upgrading from Smart Energy 1.1 to Smart Energy 2.0), configuration or log. When the device decides to send the Query Device Specific File Request command is manufacturer specific. However, one example is during upgrading from SE 1.1 to 2.0 where the client MAY have already obtained new SE 2.0 image and now needs new SE 2.0 security credential data.

The fields included in the payload helps the upgrade server in obtaining or creating the right file for the client.

11.13.10.4 Effect on Receipt

The server takes the client's information in the command and either obtain the file via the backend system or create the file itself. Details of how the file is being obtained or created is manufacturer specific and outside the scope of this document. The device specific file SHALL follow OTA upgrade file format (section 11.3) and SHALL have Device Specific File bit set in OTA header field control. Moreover, the value of the Upgrade File Destination field in the OTA header SHALL match the Request node address value in the command's field.

11.13.10.5 Handling Error Cases

In most cases all error cases resulted from receiving Query Device Specific File Request command are handled by the corresponding Query Device Specific File Response command with the exception of a few error cases described below that are handled by default response command.

11.13.10.5.1 Malformed Command

Upon reception a badly formatted Query Device Specific File Request command, for example, the command is missing one of the payload fields; the server SHALL send default response command with MALFORMED_COMMAND status to the client and it SHALL not process the command further.

11.13.10.5.2 Command Not Supported

Certain server MAY not support transferring of device specific file and the implement of Query Device Specific File Request command; in this case the server SHALL send default response with UNSUP_CLUSTER_COMMAND status.

11.13.11 Query Device Specific File Response Command

11.13.11.1 Payload Format

Figure 11-21. Format of Query Device Specific File Response Command Payload

Octets	**1**	**0/2**	**0/2**	**0/4**	**0/4**
Data Type	enum8[151]	uint16	uint16	uint32	uint32
Field Name	Status	Manufacturer code	Image type	File version	Image size

[151] CCB 2227 Status is an enumeration, not an unsigned integer

11.13.11.2 Payload Field Definitions

11.13.11.2.1 Query Device Specific File Response Status

Only if the status is SUCCESS that other fields are included. For other (error) status values, only status field SHALL be present.

11.13.11.2.2 Manufacturer Code

The value SHALL be the one received by the server in the Query Device Specific File Request command. See Chapter 2 for detailed description.

11.13.11.2.3 Image Type

The value SHALL be the one received by the server in the Query Device Specific File Request command. See section 11.4.2.6for detailed description.

11.13.11.2.4 File Version

The file version indicates the image version that the client is required to download. The value SHALL be the same as the one included in the request. See section 11.4.2.7 for more description.

11.13.11.2.5 Image Size

The value represents the total size of the image (in bytes) including all sub-elements. See section 11.4.2.10 for more description.

11.13.11.3 When Generated

The server sends Query Device Specific File Response after receiving Query Device Specific File Request from a client. The server SHALL determine whether it first supports the Query Device Specific File Request command. Then it SHALL determine whether it has the specific file being requested by the client using all the information included in the request. The upgrade server sends a Query Device Specific File Response with one of the following status: SUCCESS, NO_IMAGE_AVAILABLE or NOT_AUTHORIZED.

A status of NO_IMAGE_AVAILABLE indicates that the server currently does not have the device specific file available for the client. A status of NOT_AUTHORIZED indicates the server is not authorized to send the file to the client.

11.13.11.4 Effect on Receipt

A status of SUCCESS in the Query Device Specific File response indicates to the client that the server has a specific file for it. The client SHALL begin requesting file data using the Image Block Request or Image Page Request command with a field control value set to 0x01 and include its IEEE address. A ZED client MAY choose to change its wake cycle to retrieve the file more quickly.

If the client receives the response with status of NOT_AUTHORIZED, it MAY perform discovery again to find another upgrade server. The client MAY implement an intelligence to avoid querying the same unauthorized server.

11.13.11.5 Handling Error Cases

Query Device Specific File Response command SHALL have disable default response bit set. Hence, if the command is received successfully, no default response command SHALL be generated. However, the default response SHALL be generated to indicate the error cases below.

11.13.11.5.1 Malformed Command

Upon reception a badly formatted Query Device Specific File Response command, for example, the command is missing one of the payload field, other payload fields are included when the status field is not SUCCESS, the manufacturer code included in the command does not match that of the device or the image type value included in the command does not match that of the device; the client SHOULD ignore the message and SHALL send default response command with MALFORMED_COMMAND status to the server.

11.14 Multiple Files Required for a Bootload

ZigBee devices MAY require multiple boatload files in order to be upgraded correctly. These files often correspond to multiple embedded chips contained within the physical device that have separate firmware images to run them.

A device has a number of options for managing these files depending on its own internal configuration or dependencies. This section describes the three main options:

11.14.1 Single OTA File with multiple sub-elements

One of the simplest mechanisms to support multiple firmware images is to bundle all the images into a single OTA file. Within the OTA file each firmware image could be noted with a different sub-element tag indicating the module it is designated for. The advantage of this system is that it allows for a single OTA client to request a single OTA file from the server that contains all the upgrade data it needs. Management of the multiple firmware images is handled internally by the device.

Typically, a manufacturer would put all of the firmware images used by the device into the image and upgrade all modules at the same time. In that case the device manufacturer would need a download storage space (e.g. an external EEPROM) big enough to hold an OTA image that contained all the firmware images for all the modules.

The OTA client reports only the overall upgrade status regardless of how many internal modules are being manipulated. The OTA client’s attributes reflect only the single OTA *Image Type ID*, *CurrentFileVersion, DownloadedVersion,* and *ImageUpgradeStatus* attributes.

11.14.2 Separate OTA Files Upgraded Independently

Another method that can be used is to have each upgradeable module within the physical device request bootload images from the OTA server separately. In this case a module would report the same manufacturer ID but a different image type ID. The modules would operate on separate endpoints to properly report the attributes about the current state of that module’s upgrade cycle (*ImageUpgradeStatus*) as well as the version number it is running (*CurrentFileVersion*) and downloading (*DownloadedFileVersion*). As each module completed a download they would separately request permission to finish the upgrade via the *Upgrade End Request command*.

During the manufacturer specific part of the upgrade, it is possible that the OTA client endpoint undergoing the upgrade, or even the entire ZigBee NWK layer, MAY not be accessible over-the-air. Once the upgrade is complete the endpoint’s client attributes reflecting the new version would be updated.

Manufacturers are free to choose different versioning schemes for each image type used by the physical device and decide when to release updates for each module. However, in general it is assumed that each module can be upgraded independently of the others. Each OTA file would need to be given to the OTA server and managed separately.

Though each module operates independently it is certainly possible that specific, shared resources MAY preclude multiple simultaneous downloads or upgrades. For example, if the device has a single EEPROM that can store only one download image at a time, then only one OTA client MAY be downloading or updating. Other OTA clients on other endpoints corresponding to other modules would have to wait until the required resources are free for it to use.

11.14.3 Multiple OTA Files Dependent on Each Other

The last method a device might use to handle upgrading separate modules in the physical device is to use multiple OTA files that have a dependency on each other. In this case the OTA client would sequentially download and apply each OTA file before going to the next one.

This method might be used in the case where a single OTA file containing all the OTA images is not possible because the device does not contain a storage space big enough to hold all the module firmware images. Additionally, each module cannot operate independently due to an internal device restriction.

The details of the dependencies within the OTA files are specific to the manufacturer of the device. For example, if the device required the OTA file for Image Type 7 before it received the OTA file for Image Type 3, the device must manage this.

After each OTA file has been downloaded and processed the OTA client SHALL send an Upgrade End Request command with a status of REQUIRE_MORE_IMAGE. It SHALL then download and process the next file. In this case the act of "processing" is manufacturer specific; it MAY or MAY NOT involve upgrading the internal component. During each OTA file download the OTA client SHALL update its attributes to reflect the module that is being upgraded. For example, the Image Type ID, CurrentFileVersion, DownloadedFileVersion SHALL bet set to the values of the internal module that the OTA client is processing an upgrade image for.

Upon completion of the download for all modules the OTA client SHALL send an Upgrade End Request command with a status of SUCCESS. The OTA server has the ability to delay or abort the final upgrade via the normal mechanisms.

11.15 OTA Upgrade Cluster Management

This section provides ways for the upgrade server to monitor and manage the network-wide OTA upgrade process. It is important to realize that the server cannot reliably query the upgrade status of the sleepy devices.

11.15.1 Query Upgrade Status

Server MAY send ZCL read attribute command for Image Upgrade Status attribute on the client devices. The attribute indicates the progress of the client's file download as well as its upgrade progress. The server MAY want to make sure that all clients have completely downloaded their new images prior to issuing the Upgrade End Response command.

A client SHALL only download a single file at a time. It SHALL not download a second file while the first file download is incomplete. This insures that the values in the client's attributes can be correlated to a single download instance.

11.15.2 Query Downloaded ZigBee Stack and File Versions

The server MAY send ZCL read attribute command to a client to determine its downloaded ZigBee stack version and file version. The server SHOULD make sure that the client has downloaded the correct image prior to issuing the Upgrade End Response command.

11.15.3 Rate Limiting

The OTA Upgrade Cluster server can rate limit how quickly clients download files by setting the *MinimumBlockPeriod* attribute[152]. This feature is only available if the client supports the attribute, and the server supports this optional feature. Client support can be determined by requesting the *MinimumBlockPeriod*[153] attribute from the client, or if the Image Block Request message contains the MinimumBlockPeriod[154] field.

The server has the ability to set the attribute while the client is downloading by responding to any Image Block Request with an Image Block Response with a status of WAIT_FOR_DATA. The Image Block Response SHALL include the Block Request field with the new delay desired by the server for all the client's subsequent requests. Upon receipt of the Image Block Response the client will record the new value in its local *MinimumBlockPeriod* attribute and use it for the rest of the download.

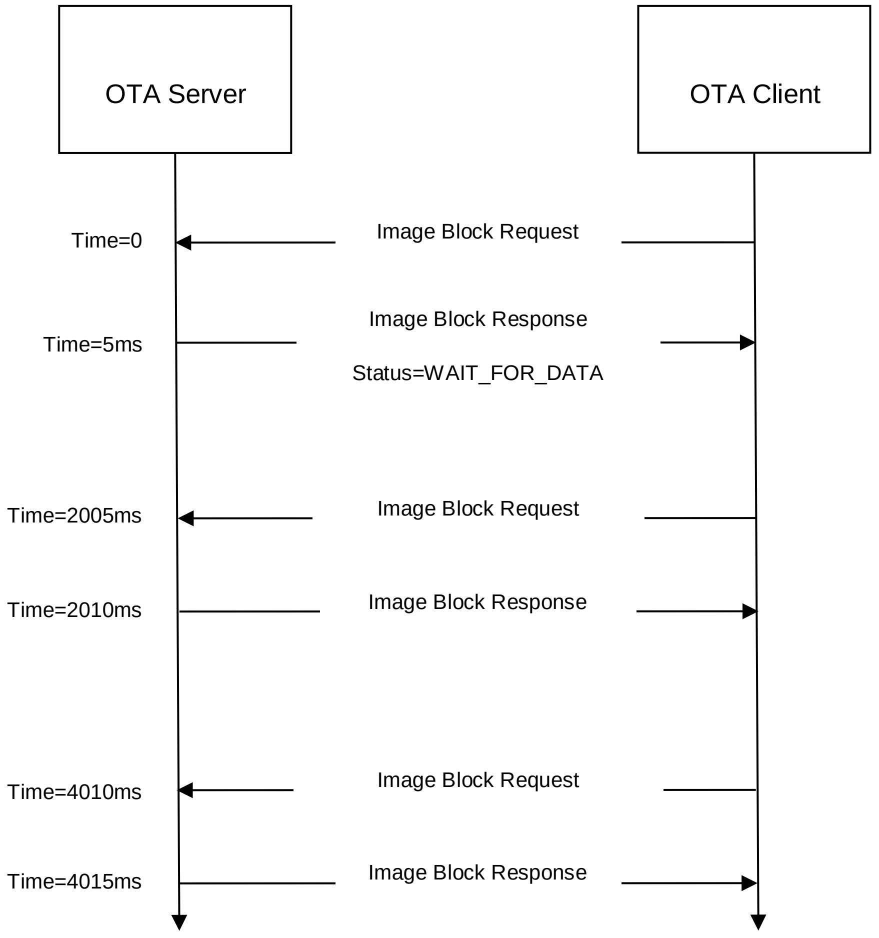
The server can change the download delay of the client multiple times over the course of the download based on whatever criteria it deems appropriate. For example, if the server detects only 1 client is downloading, it could allow that client to download at full speed (*MinimumBlockPeriod* = 0), but if other clients simultaneously start downloads it could limit all clients to 1 Image Block Request every 500 milliseconds[155]. Alternatively, it could give higher priority to certain clients to download their upgrade image and let them download at full speed, while slowing down other clients.

The *MinimumBlockPeriod* attribute is a minimum delay. The client MAY request data slower than what the server specifies (i.e. with a longer delay). Sleeping end devices MAY do this normally to conserve battery power.

Below is a diagram showing how the rate limiting process generally works.

[152] CCB 2307 align attribute names with field names
[153] CCB 2307 align attribute names with field names
[154] CCB 2307 align attribute names with field names
[155] CCB 2398 *MinimumBlockPeriod* is milliseconds (not seconds)

Figure 11-22. Rate Limiting Exchange



11.15.4 Current Time, Request Time, and MinimumBlockPeriod

When a server sends an Image Block Response with a status of WAIT_FOR_DATA, it can delay the client's next Image Block Request. This can be done persistently for all subsequent requests, or temporarily as a onetime delay.

The onetime delay can be created by setting the Current Time and Request Time fields as described in section 11.13.8.2.8. This might occur if the server does not immediately have access to the block of the upgrade image requested by the client, and the server must fetch the block from another location.

The persistent delay can be enabled by setting the MinimumBlockPeriod as described in section 11.13.6.2.8 however this only works if the client and server support this functionality.

11.16 OTA Upgrade Process

Once a device has completely downloaded the image and returned a status of SUCCESS in the Upgrade End Request, it SHALL obey the server's directive based on when it SHOULD upgrade. However, there are many failure scenarios where this MAY not be possible. In such failure case, the device SHOULD attempt to contact the server and determine what SHOULD be done, but if that has failed as well, then it MAY apply its update without an explicit command by the server.

After receiving an Upgrade End Response from the server the client will apply the upgrade according to time values specified in the message. If the response directs the device to wait forever, it SHALL periodically query the server about when it SHOULD apply the new upgrade. This SHALL happen at a period no more often than once every 60 minutes. If the server is unreachable after 3 retries, the device MAY apply the upgrade (see section 11.11.4 for further details).

The client does not need to persistently store the time indicating when to apply the upgrade. If the client feels that it has lost connection to the upgrade server, it SHALL first try to rediscover the upgrade server perhaps by rejoining to the network and performing network address discovery using the stored UpgradeServerID attribute. Once the server is found, the client SHALL resend an Upgrade End Request command with a status of SUCCESS to the server, including the relevant upgrade file information. The server SHALL send it a response again indicating when it SHOULD upgrade. If the device is unable to communicate to the upgrade server or it cannot synchronize the time, it MAY apply the upgrade anyway.

When the time comes for the client to upgrade, the device SHOULD begin the manufacturer specific method to upgrade its image. The upgrade MAY involve one or more hardware resets. Once the device has completed the upgrade it SHOULD be able to reinitialize itself and start communicating on the network again. Previous network information such as channel, power, short pan id, extended pan id SHOULD be preserved across the upgrade.

11.17 Application Standard Specific Decisions

Below are the decisions that each application standard needs to make in order to ensure successful OTA upgrade of devices in the network.

- The following are security considerations that SHOULD be taken into account when using this cluster.
  - Whether image signatures will be used to sign the OTA upgrade file. If a signature is used, what type of image signature will it be (example: ECDSA).
  - What encryption will be used during the transport of OTA data
- Whether to use offset or UTC time in Image Block Response and Upgrade End Response commands. Refer to sections 11.11.4 and 11.13.9 for more details. If the application standard does not specify which type of (OTA upgrade) time to support, it is default to using the offset time since it does not require an implementation of ZCL time cluster. Once the application standard has decided which type of time to support, only that type of time SHALL be used consistently across the OTA upgrading. The standard SHALL avoid using both types of time simultaneously to avoid any confusion and inconsistency between the two time values.

Other application standard wide decisions that SHOULD be answered are:

- How often the OTA client SHALL discover the OTA server until it finds one that is authorized to do the upgrade.
- How often the ZED client SHALL query the OTA server for new OTA upgrade image.
- How often the ZED devices SHALL query for image data.

11.17.1 SE Profile Standard: OTA Upgrade from SE 1.x to SE 2.0

The definition of SE Profile 2.0 is currently still being worked on by the ZigBee SE group. However, it is suggested that in order to successfully upgrade a device from SE 1.x to SE 2.0, such process MAY involve transferring new security data over-the-air from the server to the client device. OTA Upgrade cluster has provided a set of commands that MAY be used to obtain such security data. The security data will be requested separately by the client using Query Device Specific File Request command. The data is sent from the server to the client as an OTA upgrade file via similar set of commands used to request firmware image. This OTA security file will be specific to a particular client device.

A client SHOULD request new security data necessary for SE Profile 2.0 via Query Device Specific File Request command. After obtaining the security data file, the server will include the file information in the Query Device Specific File Response command in response to the client’s request. Upon reception the response, the client then SHALL obtain the file via Image Block or Page Request command. Query Device Specific File Request and Response commands are described in sections 11.13.10 and 11.13.11 respectively.

11.18 OTA Upgrade Recovery

Each manufacturer is encouraged to implement a recovery method that SHOULD be used to recover the node in a case when the OTA upgrade fails. The recovery method is particularly important in a case where the device MAY not be able to communicate to the server over-the-air. The actual recovery implementation is manufacturer specific; however, some of the options are discussed in this section.

One option for recovery method is the ability for the application bootloader to swap the images between its external flash and its internal flash, rather than just overwriting the internal with the external. A sample use case is where the upgraded device is functional enough to receive a message, but broken enough to not be able to initiate OTA upgrade process again. A manufacturer specific command MAY be sent from the server to notify the device to revert back to its previous image.

In a case where the device is no longer able to communicate to the server over-the-air; the application bootloader could revert to the previous image via a button press on power up.

CHAPTER 12 TELECOMMUNICATION

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

12.1 General Description

12.1.1 Introduction

The clusters specified in this chapter are for use typically in telecommunication applications but may be used in any application domain.

12.1.2 Cluster List

This section lists the clusters specified in this chapter and gives examples of typical usage for the purpose of clarification. The clusters specified in this chapter are listed in Table 12-1.

Table 12-1. Telecom Cluster List

ID	Cluster Name	Description
0x0900	Information	Commands and attributes for information delivery
0x0905	Chatting	Commands and attributes for sending chat messages
0x0904	Voice Over ZigBee	Commands and attributes for voice receiving and transmitting

12.2 Information

12.2.1 Scope and Purpose

This section specifies the Information cluster, which provides commands and attributes for information delivery service on ZigBee networks and also specifies three types of special nodes on which this cluster works. One of the nodes, Information Node (IN) is a node which provides information contents in both pull-based and push-based information delivery to a mobile terminal. The contents may have links to other contents and thus they may be organized in a structure. Mobile Terminal (MT) is the one of special nodes and is used by an end-user who looks into information from the information node. The other node is Access Point (AP) which updates contents stored in Information nodes and has a role of gateway connected to the operator network. It is also assumed to be a ZigBee coordinator which forms a network with Information nodes and Mobile Terminals. Access point may have a function of Information Node.

This section should be used in conjunction with the Foundation Chapter, which gives an overview of the library and specifies the frame formats and general commands used therein.

Information Delivery Service in this document is considered ‘Pull-based delivery’ and ‘Push-based delivery’. Both methods are provided by a single cluster, the information cluster.

Figure 12-2 shows typical usage of the cluster. This cluster may use Partition cluster.

12.2.1.1 Data Structure of Contents Data

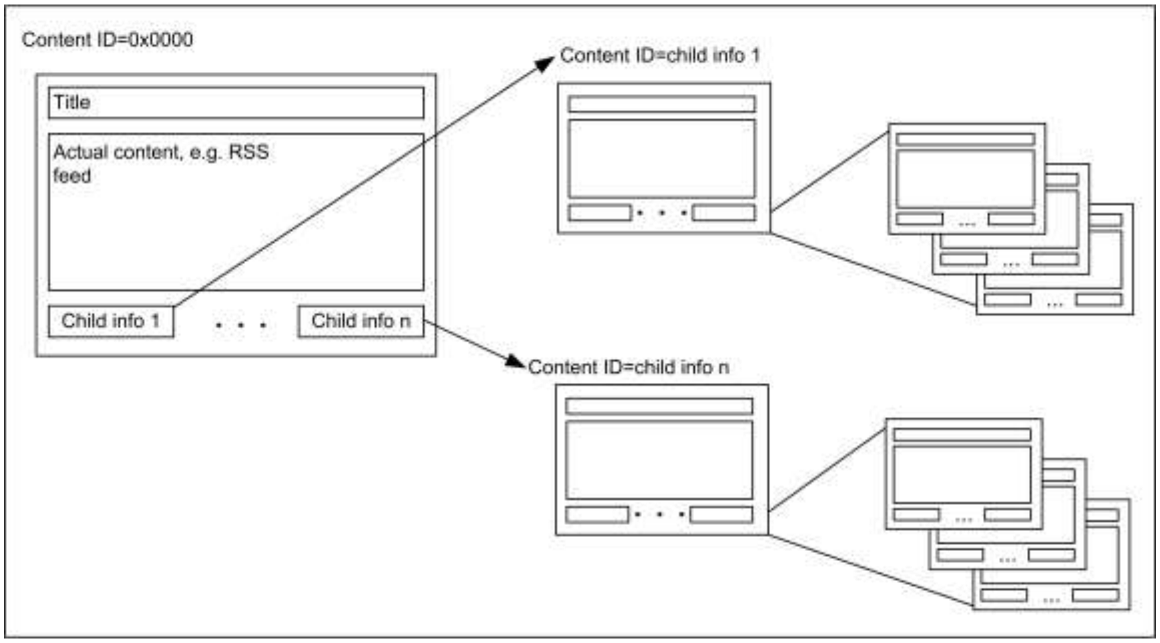
Typical data structure of contents data is as illustrated in Figure 12-1. Each content data has its Content ID, which is used when the client cluster requests content to the server cluster.

A content data includes 'the title strings', 'actual content', 'number of child contents' and 'children's content IDs'.

To let each content data have its children content data makes it enables to organize list-structure or tree-structure and obtain a hierarchical content data structure, and a user who uses information delivery can request information along to child information links.

To obtain the first contents ID, there are methods, reading server attribute 'Root ID', using ID sent by out-of-band like via GPRS network and using ID provided by another telecom application clusters (ex. Payment or gaming).

Figure 12-1. Typical Content Data Structure



12.2.2 Cluster List

A cluster specified in this document is listed in Table 12-2.

Server cluster is expected to be implemented in the Information Node. Client cluster (including functions related to contents provisioning) is expected to be implemented in the Mobile Terminal. A part of commands of client cluster, update command and configuration commands are expected to be implemented in the Access Point. The Access Point may have functionality of the Information node and it has server cluster in that case. Some user specific content is provided with processing user-side information, defined as a preference. If a preference needs to be processed not in the Information Node but in an Access Point or in a server beyond the Access Point as a gateway, information indicates the Access Point's ID so that the Mobile Terminal can switch to access it (as illustrated in Figure 12-3) or the Information Node acts as proxy and access the Access Point with client function to forward preference, commands to the Access Point and contents to the Mobile Terminal (as illustrated in

Figure 12-2).

Table 12-2. Clusters Specified for the Information Delivery

Cluster Name	Description
Information cluster	Attributes and commands for providing Information service to a ZigBee device.

Figure 12-2. Typical Usage of the Information Cluster

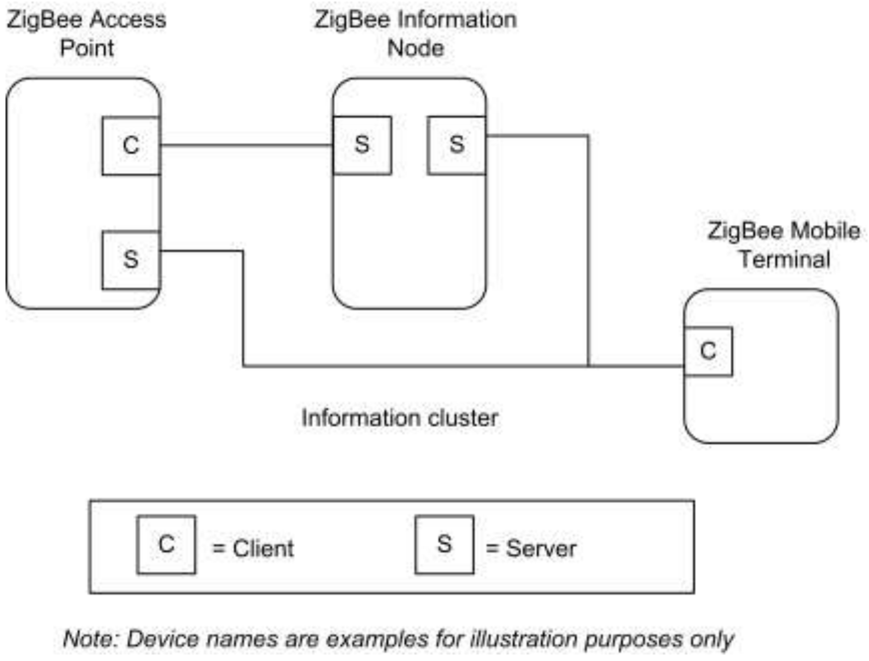
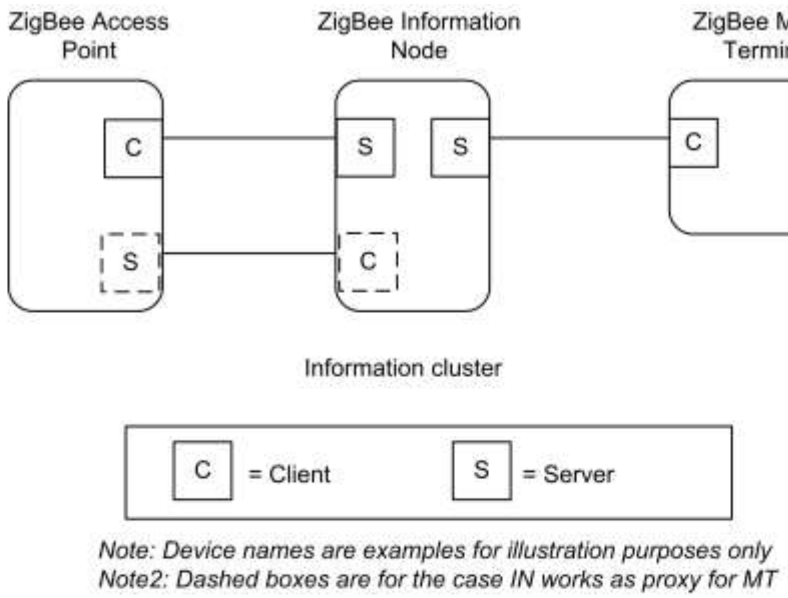


Figure 12-3. Typical Usage of the Information Cluster – with Proxy Function


ZigBee Access Point
ZigBee Information Node
ZigBee Mobile Terminal
C
S
S
S
C
C
Information cluster
C = Client
S = Server
Note: Device names are examples for illustration purposes only
Note2: Dashed boxes are for the case IN works as proxy for MT

12.2.3 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides attributes and commands for Information Delivery Service.

12.2.3.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added;CCB 1811 1812 1821

12.2.3.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	TELIN	Type 2 (server to client)

12.2.3.3 Cluster Identifiers

Identifier	Name
0x0900	Information (Telecom)

12.2.4 Server

The Information Node (IN) has a server cluster which provides information delivery service. A client cluster in Mobile Terminal (MT) requests information and IN responds with requested contents on pull-based delivery. Besides, cluster can provides push-based delivery so the server cluster in the IN sends contents to client cluster in the MT (if properly configured).

Content may have links to the other contents. A link is called as child information in this document and it is represented as a ContentID. Contents can be organized in tree-structure.

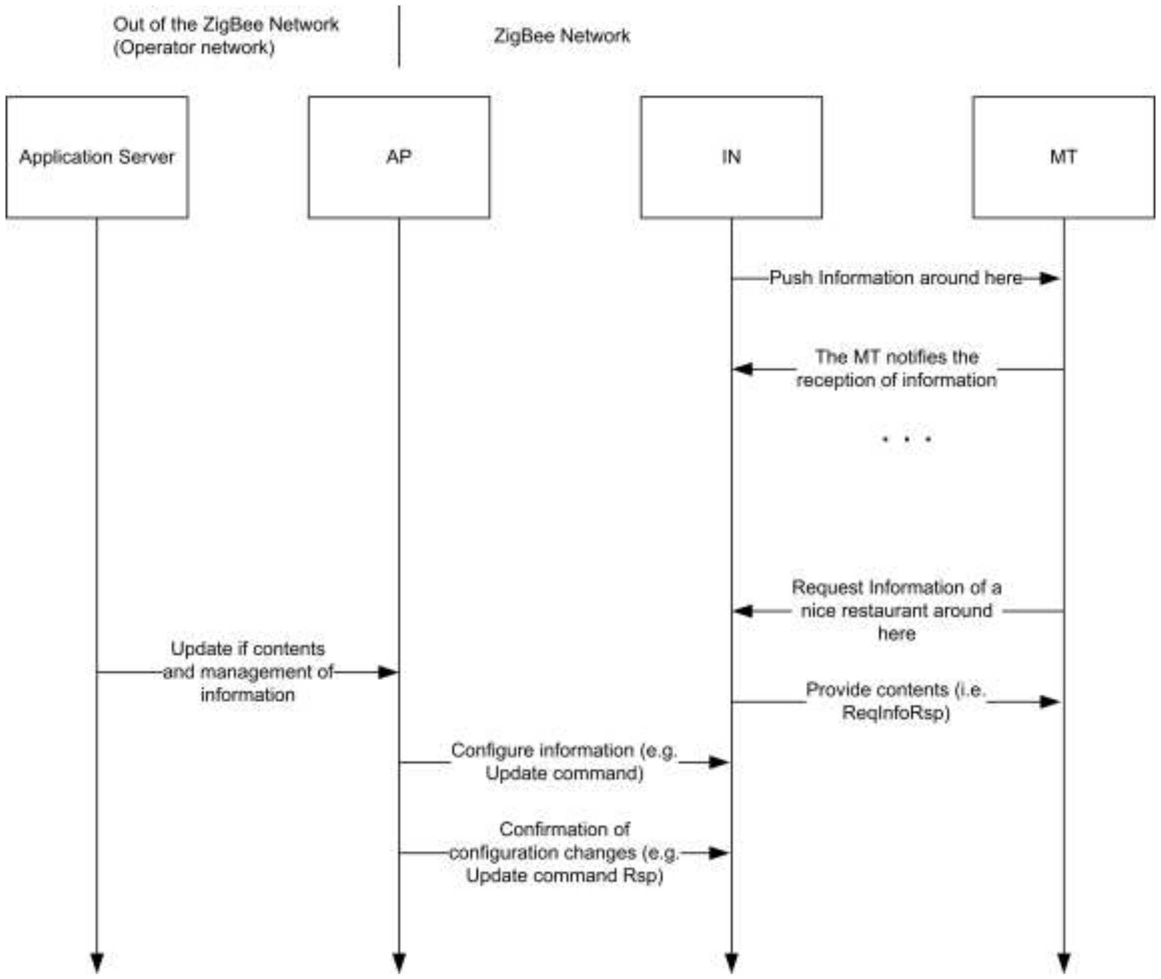
Content may be one of three explicitly specified types: octet strings, character strings or RSS feed, so that the browser in the MT can understand easily what content it should access.

Cluster also provides such function that the client cluster in the AP can update contents and delete them in the IN.

Preference is used for carrying user-side information to let the IN provide user specific contents based on the user-side information. Contents may be modified along with that information on the IN. An example scenario of Information cluster is illustrated in Figure 12-4.

A preference may be processed not in an IN but in an AP or in a server beyond the AP as a gateway. In that case, the IN needs to have client function to forward preference, commands and contents as proxy for MT (Forwarding scenario) or the IN needs to inform the MT to switch its access from the IN to the AP (Redirection scenario). The Cluster supports both scenarios. If the preference, commands and contents are forwarded by the IN between the MT and the AP, they may be just relayed transparently through the IN, or they may be processed by the IN. The IN may process the preference before forwarding it, and may process the stored preference together with the contents to create the customized contents after receiving the response from the AP, then sending the contents to the MT.

Figure 12-4. An Example Sequence


Out of the ZigBee Network
(Operator network)
ZigBee Network
Application Server
AP
IN
MT
Push Information around here
The MT notifies the reception of information
. . .
Request Information of a nice restaurant around here
Update if contents and management of information
Provide contents (i.e. ReqInfoRsp)
Configure information (e.g. Update command)
Confirmation of configuration changes (e.g. Update command Rsp)

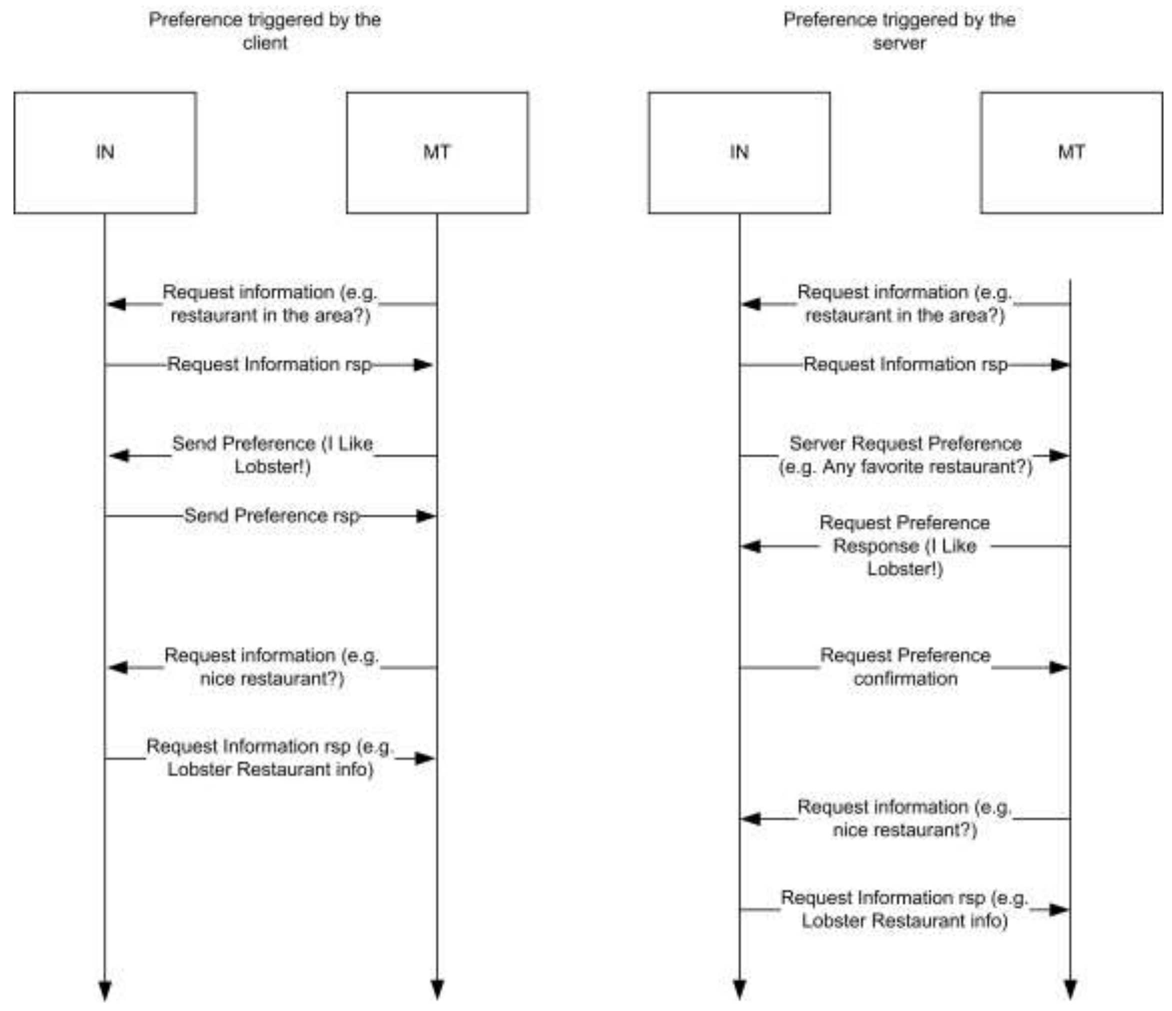
Pull-based service is expected to work as follows:

1. It provides decentralized contents distributed by Update command from the central node (the AP). (e.g., tree-structure contents distribution, specific permission to peep the contents to the authorized user)

2. Advanced application program provides service in conjunction with the other functions like a Location cluster, or the preference data from the MT. (e.g., direction service based on the location information of user, push service matching individual attribute – invitation of a test drive of new car to men in thirties which hobby is driving or etc.)
3. Hybrid service of the item a. and b.

The preference format is application dependent and is used by the service like the item b. Of course the application uses the preference shall have the ability to parse it. If the application doesn't have the ability, it shall report it that. The cluster provides a status code to inform it. For example, let's assume the IN provides service like item a. and the AP connected a network out of the ZigBee and an application server is deployed there. First the MT access to the IN to get general contents for the all of users. The IN can provide simple contents to the MT. Second, the MT request a content – good restaurant to the IN, which content is indicated to redirect to the AP. The MT switch its access to the AP. The AP requests preference to the MT to get user-side information, favorite food in the example. The AP sends the preference to the application server and it replies with the food restaurant which the user likes. Thus the AP can provide the content modified along with user-side information – the restaurant which provides he likes – to the MT. The example illustrated in Figure 12-5.

Figure 12-5. Preference Scenarios (Triggered by the Client or by the Server)


Preference triggered by the client
IN
MT
Request information (e.g. restaurant in the area?)
Request Information rsp
Send Preference (I Like Lobster!)
Send Preference rsp
Request information (e.g. nice restaurant?)
Request Information rsp (e.g. Lobster Restaurant info)
Preference triggered by the server
IN
MT
Request information (e.g. restaurant in the area?)
Request Information rsp
Server Request Preference (e.g. Any favorite restaurant?)
Request Preference Response (I Like Lobster!)
Request Preference confirmation
Request information (e.g. nice restaurant?)
Request Information rsp (e.g. Lobster Restaurant info)

12.2.4.1 Dependencies

None

12.2.4.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 12-3.

Table 12-3. Information Cluster Attribute Sets

Attribute Set Identifier	Description
0x000	Node Information
0x001	Contents Information
0x002 – 0xfff	Reserved

12.2.4.2.1 Node Information Attribute Set

The Node Information attribute set contains the attributes summarized in Table 12-4.

Table 12-4. Node Information Attribute Set

Id	Name	Type	Range	Access	M/O
0x0000	*NodeDescription*	string		R	M
0x0001	*DeliveryEnable*	bool	0x00 – 0x01	R	M
0x0002	*PushInformationTimer*	uint32		R	O
0x0003	*EnableSecureConfiguration*	bool	0x00 – 0x01	R	M

12.2.4.2.1.1 NodeDescription Attribute

This *NodeDescription* Attribute holds strings which indicate what Information Delivery service is available so that an end-user can select and distinguish this service.

12.2.4.2.1.2 DeliveryEnable Attribute

The *Delivery Enable* attribute is Boolean and indicates whether the cluster is able to communicate with the other nodes. It is a read only attribute but it can be changed by using the Configure Delivery Enable command. If it is set to TRUE (0x01), the Information cluster is able to manage the following commands: Request Information, Push Information Response, Send Preference and Request Preference Response.

12.2.4.2.1.3 PushInformationTimer Attribute

The *Push Information Timer* is an Unsigned 32-bit integer and indicates whether the cluster is able to send Push Information command and the time between those commands. It is a read only attribute but it can be changed by using the Configure Push Information timer command. If this attribute is set to 0, then the automatic Push Information is disabled, otherwise the value is considered as an interval (in milliseconds) that elapses between Push Information commands. If this attribute is set to 0, it's still possible for the device to push information triggered by an event such as button being pushed.

12.2.4.2.1.4 EnableSecureConfiguration Attribute

The Enable Secure Configuration attribute is a Boolean and indicates whether an application layer security is required in order to process the configuration commands: Update, Delete, Configure Delivery Enable, Configure Set Root ID, Configure Node Description, Configure Push Information timer. If this attribute is set to TRUE, then server side of the cluster need to use application link keys for processing those commands. If FALSE, then all the commands can be processed without using link keys.

12.2.4.2.2 Contents Information Attribute Set

The Node Information attribute set contains the attributes summarized in Table 12-5.

Table 12-5. Contents Information Attribute Set

Identifier	Name	Type	Range	Access	M/O
0x0010	*NumberOfContents*	uint16	0x0000 - 0xFFFF	R	O
0x0011	*ContentRootID*	uint16	0x0000 - 0xFFFF	R	O

12.2.4.2.2.1 NumberOfContents Attribute

This attribute holds the total number of contents which this server node has. It should reflect the result of updating command by AP. This attribute holds the total number of contents which this server node has. If the number is more than 0xffff, this attribute shall be set to 0xffff.

12.2.4.2.2.2 ContentRootID Attribute

This attribute holds root Content ID of octet strings, character string and RSSFeed Contents. *ContentRootID* is a start pointer so that user can access variety contents. If this attribute doesn’t exist, there are no contents. 0xffff for this attribute means it is not specified yet.

12.2.4.3 Commands Received

The received command IDs for the information cluster are listed in Table 12-6. Please notice that at least one of the commands shall be implemented though they are defined as optional.

Table 12-6. Received Command IDs for the Information Cluster

Id	Description	M/O	Command Type
0x00	Request Information	M	*Operation*
0x01	Push Information Response	M	*Operation*
0x02	Send Preference	O	*Operation*
0x03	Request Preference Response	O	*Operation*
0x04	Update	O	*Configuration*
0x05	Delete	O	*Configuration*
0x06	Configure Node Description	O	*Configuration*
0x07	Configure Delivery Enable	O	*Configuration*
0x08	Configure Push Information Timer	O	*Configuration*
0x09	Configure Set Root ID	O	*Configuration*

12.2.4.3.1 Request Information Command

This is a command requesting information as a list, as a content of text strings and as an RSS feed from mobile terminal to the Information Node or to the Access Point. An Information Node (or an Access Point) that receives this command shall reply by Request Information Response Command with requested information to the sender of this command. It specifies how to indicate content By the ‘Inquiry Type’ and also specifies what data type of content is requested by the ‘Data Type ID’. For example, in pull scenario, MT gets contents list, sending this command (e.g., Inquiry ID = ‘Request by depth’) and receiving Request Information Response Command with the list of titles. By another Request Information Command indicating contents ID, MT can get an individual content.

12.2.4.3.1.1.1 Frame Format

The Request Information command shall be formatted as illustrated in Figure 12-6.

Figure 12-6. Payload Format of Request Information Command

Octets	1	1	Variable
Data Type	enum8	map8	See 12.2.4.3.1.2
Field Name	Inquiry ID	Data Type ID	Request Information Payload

Inquiry ID shall be set as one of IDs listed in Table 12-7.

Data Type ID indicates what type of contents the response command requires. It shall be formatted by combination of bitmasks described in Table 12-8. A bit for ‘Title’ indicates the request requires ‘Title strings’ and it can be combined other type of contents. Flagging ‘Title’ bit means a request title be attached and the other bits used for filter. If ‘Title’ bit, ‘Octet’ bit and ‘RSS’ bit are flagged, that means request is “Octet content attached title and RSS content attached title are required.” In the case that the only ‘Title’ bit is flagged, the request means “Just titles are required.” Please notice that all the contents shall maintain a title in the local database.

Table 12-7. Inquiry ID

Inquiry ID	Description	M/O
0x00	Request a content by a content ID	M
0x01	Request contents by multiple IDs	O
0x02	Request all	O
0x03	Request by depth	O

Table 12-8. Data Type IDs

Data Type ID	Bit Mask	Description
0x01	0000 0001	Title
0x02	0000 0010	Octet String
0x04	0000 0100	Character String

Data Type ID	Bit Mask	Description
0x08	0000 1000	RSS Feed
0x1*X* – 0xf*X*	-	Reserved

12.2.4.3.1.2 Request Information Payload

Request Information Payload changes along with Inquiry ID listed in Table 12-7. Payload formats for each Inquiry ID are described following sections.

12.2.4.3.1.3 Inquiry ID

12.2.4.3.1.3.1 Format for Request a Content by a Content ID

The command with this ID requests a single content by a Content ID. Format is illustrated in Figure 12-7.

A server shall respond Request Information Response command with a content indicated by the content ID.

Figure 12-7. Payload Format for Request a Content by a Content ID

Octets	2
Data Type	uint16
Field Name	Content ID

12.2.4.3.1.3.2 Format for Request Contents by Multiple IDs

The command with this ID requests several contents by indicating several content IDs. It shall be formatted as illustrated in Figure 12-8.

A server shall respond Request Information Response command with contents indicated by content IDs.

Figure 12-8. Request Information Payload for Request Contents by Multiple IDs

Octets	2	2	…	2
Data Type	uint16	uint16		uint16
Field Name	Content ID 1	Content ID 2	…	Content ID *n*

12.2.4.3.1.3.3 Format for Request All

The command with this ID requests all contents. No payload format is specified and it should be empty.

A server shall respond Request Information Response command with all contents.

12.2.4.3.1.3.4 Format for Request by Depth

Upon receipt of the command with this ID, server shall reply Request Information Response command with concatenated contents indicated by Start ID and Depth. Request Information Payload format for this ID is specified in Figure 12-9.

Start ID field holds content ID for starting point to retrieve structured contents.

Depth field holds how many levels to request from Start ID tracing child information. If a depth equals to 0x00, the requested content should be single content of Start ID itself.

Server shall provide concatenated contents, which needs a prevention of duplication induced by the loop of links. (For example, if the content has a child content which child ID refers its parent (A→B, B→A), there is a loop. If the requester indicates 2 for the depth and requests content "A", searching child information would be like as A→B→A. However only content A and B should be carried in this case).

Figure 12-9. Request Information Payload for Request by Depth

Octets	2	1
Data Type	uint16	uint8
Field Name	Start ID	Depth

12.2.4.3.2 Push Information Response Command

This command is used by the client to notify the reception of the data carried by Push Information Command, and it is used by the server to confirm if it is correctly stored or not into MT. This command shall not be used if the Push Information Command is sent by broadcast. It is to prevent explosion of response.

Payload format shall be as illustrated in Figure 12-10.

Figure 12-10. Payload Format of Push Information Response Command

Octets:	**2**	**1**	**…**	**2**	**1**
Field:	Notification 1		…	Notification *n*	
	Content ID 1	Status Feedback 1		Content ID *n*	Status Feedback *n*

Notification field has two sub-fields, content ID and Status Feedback. Content ID indicates what content the notification has the status for. Status Feedback indicates the status of the reception of the content.

Possible message for Status Feedback are SUCCESS, FAILURE, MALFORMED_COMMAND, UNSUP_CLUSTER_COMMAND, INVALID_FIELD, INSUFFICIENT_SPACE, HARDWARE_FAILURE and SOFTWARE_FAILURE already specified in the enumeration lists of ZCL [R4].

12.2.4.3.3 Send Preference Command

This command carries a preference that is specific information of interest for the user, from the client to the server. Upon receipt of this command on the server, the server application may modify or change user specific contents along with preference information. The type of data put into the preference is based on the Preference Type field. Payload format for this command shall be as illustrated in Figure 12-11.

Figure 12-11. Payload Format for Send Preference Command

Octet:	2	Variable
Field	Preference Type	Preference Payload, see Table 12-9

The Preference Type determines the format of the preference Payload. All devices must support Preference Type of 0x0000.

Table 12-9. Preference Type

Preference Type	Description
0x0000	Preference is Multiple Content ID
0x0001	Preference is Multiple Octet Strings
0x0002 – 0x7fff	Reserved
0x8000 – 0xfffb	Used for Vendor Specific Format
0xfffc – 0xffff	Reserved

Figure 12-12. Payload Format for Preference Is Multiple Content ID (0x0000)

Octets	1	2	…	2
Data Type	uint8	uint16	…	uint16
Field Name	Count	Content ID 1		Content ID N (based on Count)

Figure 12-13. Payload Format for Preference Is Multiple Octet Strings (0x0001)

Octets	1	Variable (1-256)	…	Variable (1-256)
Data Type	uint8	octstr	…	octstr
Field Name	Count	Preference Data 1		Preference Data N (based on Count)

As described in Figure 12-5 there are two scenarios for the preference:

4. Preference triggered by server side (Information Node or Access Point):

- IN ←(Request Information) ← MT
- IN → (Request Information Response) → MT
- IN → (Server Request Preference) → MT
- IN ← (Request Preference Response) ← MT
- IN → (Request Preference Confirmation) → MT
- IN ← (Request Information) ← MT

- IN → (Request Information Response) → MT

5. Preference triggered by client side (e.g., Mobile terminal):

- IN ← (Request Information) ← MT
- IN → (Request Information Response) → MT
- IN ← (Send Preference) ← MT
- IN → (Send Preference Response) → MT
- IN ← (Request Information) ← MT
- IN → (Request Information Response) → MT

12.2.4.3.4 Request Preference Response Command

This command carries a preference as a response of 'Server Request Preference' command on pull-basis. Format shall be as illustrated in Figure 12-14.

Figure 12-14. Payload Format of Request Preference Response Command

Octets:	**1**	**2**	**Variable**
Field:	Status Feedback	Preference Type	Preference Payload, see Table 12-9

Status Feedback carries a message as a response to previous 'Server Request Preference' command. Possible messages are SUCCESS, FAILURE, NOT_FOUND, MALFORMED_COMMAND, UNSUP_CLUSTER_COMMAND, INVALID_FIELD, HARDWARE_FAILURE and SOFTWARE_FAILURE already specified in enumeration lists of ZCL [R1]. Besides, REQUEST_DENIED is included to these messages for this cluster specification.

12.2.4.3.5 Update Command

Server cluster in the IN which receives this command from the AP shall updates contents by the one which the command carried except that there is an error in the IN. Update command also indicates various control to the contents by the control fields. Control fields affect to all of contents carried by the Update command, so contents required to be indicated different control should be carried by another Update command.

Payload format is as illustrated in Figure 12-15.

Figure 12-15. Payload Format for Update Command

Octet	1	1	Variable
Data Type	enum8	map8	Payload Format for Multiple Content
Field Name	Access Control Field	Option Field	Contents Data

12.2.4.3.5.1 Access Control Field

Access Control Field is 8-bit enumeration and is used to indicate security level for the validation to access the contents which are carried by the Update command. All of contents carried by the Update command shall be affected by this control field. The enumeration values are listed up in Table 12-10.

Table 12-10. Value of the Access Control Field

Access Control Mode Value	Description
0x00	Free to access
0x01	Link key establishment based
0x02	Billing based
0x03 – 0xfe	Reserved
0xff	Vendor Specific

Free to access: All of the clients is permitted to access the contents without special validation.

Link key establishment based: The client to access to the IN shall be required to establish link key establishment to achieve the contents. Contents shall be encrypted by the link key.

Billing based: The client to access the contents is required to finish the Billing cluster procedure.

Vendor specific: No special method is defined in this document. The application defines it. (Out-of-box, out-of-band, etc.)

12.2.4.3.5.2 Option Field

Option Field is used for advanced indication while updating contents. Forwarding flag, Redirection flag, Overwrite update flag are defined in the current version. The 'Forwarding' flag or the 'Redirection' flag are used to indicate 'content' so that the commands of request and response related to the indicated 'content' shall be forwarded or redirected to the AP. If both 'Forward' flag and 'Redirection' flag are 1, the server cluster shall reply the INVALID_FIELD by the Update Response command.

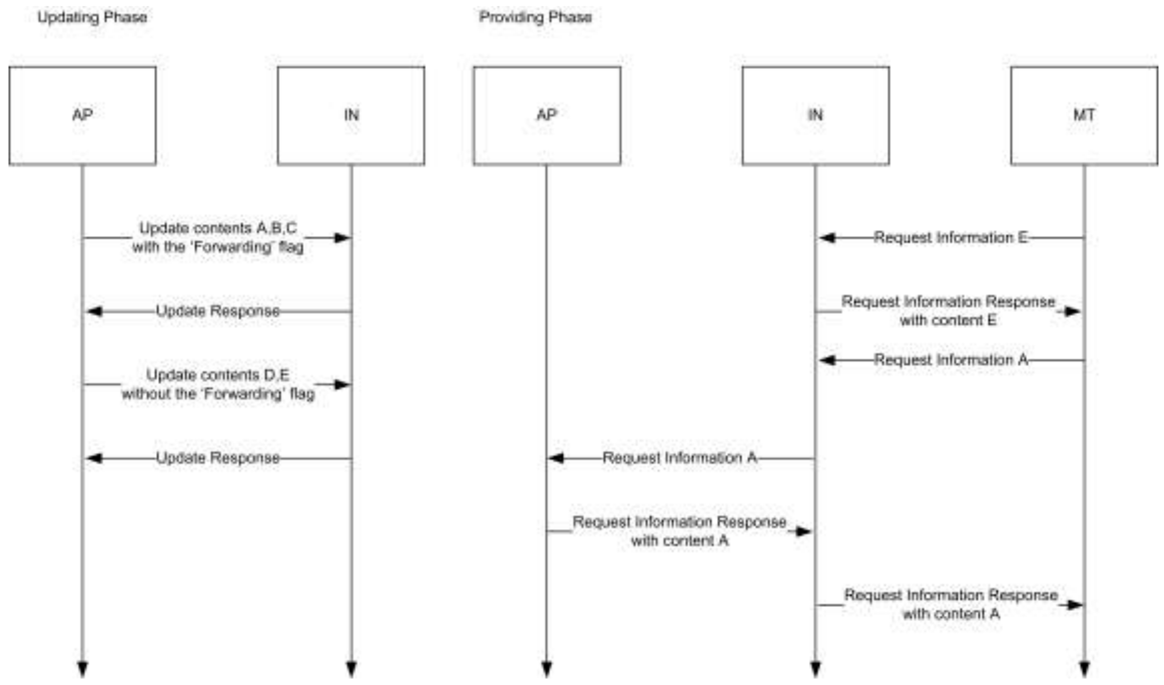
The format is as illustrated in Figure 12-16.

Figure 12-16. Format for Redirection Control Field

Bits: 1	1	1	5
Forward	Redirection	Overwrite update	Reserved

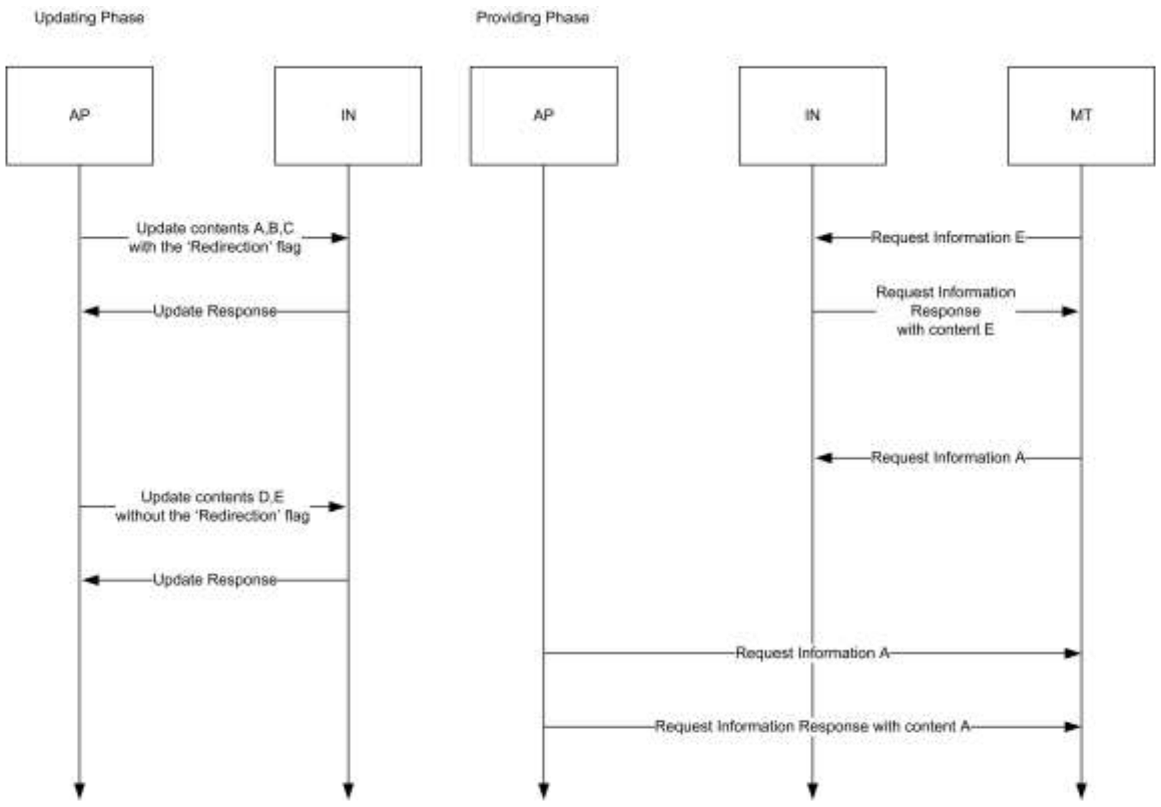
Forward flag: The Information Node is required to forward messages from the MT to the AP and message from the AP to the MT with acting as proxy. All the requests from the MT for the contents updated with this flag are forwarded to the AP. A Preference from the MT is also sent to AP if the IN has it. The AP answers the response command to the IN with requested contents and they are forwarded to the MT similarly. Figure 12-17 shows an example usage of forwarding.

Figure 12-17. An Example Sequence of Forwarding Case


Updating Phase
Providing Phase
AP
IN
AP
IN
MT
Update contents A,B,C with the 'Forwarding' flag
Update Response
Update contents D,E without the 'Forwarding' flag
Update Response
Request Information E
Request Information Response with content E
Request Information A
Request Information A
Request Information Response with content A
Request Information Response with content A

Redirection flag: A client requested the contents indicated by this flag shall receive Request Information Response command with Status feedback 'INDICATION_REDIRECTION_TO_AP'. The client is required to switch to access from the Information Node to the Access Point. This flag makes MT enable to switch access to AP automatically without user's operation (Like that user access the child content). For example, let IN have general site-dependent information and content depends on user-side information generated by a server beyond the operator network which AP is connected. Figure 12-18 shows an example usage of redirecting.

Figure 12-18. An Example Sequence of Redirecting Case


Updating Phase
Providing Phase
AP
IN
AP
IN
MT
Update contents A,B,C with the 'Redirection' flag
Update Response
Request Information E
Request Information Response with content E
Request Information A
Update contents D,E without the 'Redirection' flag
Update Response
Request Information A
Request Information Response with content A

Overwrite: For the case that the IN has already contents corresponding to the one required to Update, the command indicates if it can be overwritten or not. If it is 0b1, the contents carried by the Update command overwrites the contents which the IN has. If it is 0b0, overwriting is not permitted. In that case, the error 'FAILURE' on Update Response command is issued by the IN and the command is ignored if the IN has corresponding contents.

12.2.4.3.6 Delete Command

Server cluster in the IN which receives this command from the AP shall delete contents by the one which the command carried except that there is an error in the IN. Delete command also indicates various control to the contents by the control fields. Control fields affect to all of contents carried by the Delete command, so contents required to be indicated different control should be carried by another Delete command.

Payload format is as illustrated in Figure 12-19.

Figure 12-19. Payload Format for Delete Command

Octet	1	2	…	2
Data Type	map8	uint16	…	uint16
Field Name	Deletion Option	ContentID 1	…	ContentID *n*

12.2.4.3.6.1 Deletion Option Field

Deletion Option field enables various deletion functions. The format is as illustrated in Figure 12-20.

Figure 12-20. Format for Deletion Option Field

Bits: 1	**2-8**
Recursive	Reserved

Recursive: If it is 0b1, all the sub tree starting for the content carried by the Delete command are deleted. If it is 0b0, only the content carried by the Delete command is deleted. How the children are linked to the rest of the tree is out of scope of this document, it is application dependent.

12.2.4.3.7 Configure Node Description

Payload format for the Configure Node Description command shall be as illustrated in Figure 12-21.

Upon recipient of this command, the server cluster shall change its Node Description attribute to the value of the "Description" field in this command. The use of this specific command guarantees that Node Description attribute can be reconfigured only when Delivery Enable attribute is set to TRUE. Upon reception of this command the recipient will reply with Default Response with status field equal to SUCCESS (if the requester set the Disable Default response bit of ZCL header to 0). The Configure Node Description command will be acknowledged with Default Response with status field equal to NOT_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e., Enable Secure Configuration attribute set to TRUE) while the sender didn't use the proper link key for sending the configuration commands.

Figure 12-21. Payload Format for Configure Node Description Command

Octets	**Variable**
Data Type	string
Field Name	Description

12.2.4.3.8 Configure Delivery Enable

Payload format for the Configure Delivery Enable command shall be as illustrated in Figure 12-22.

Upon recipient of this command, the server side of the Information cluster should set the value present in the 'Enable flag' field into the Delivery Enable attribute. Note that, if Enable Secure Configuration attribute is set to TRUE (0x01) this command should be handled only whether the entity sending this message will have previously set up its link key with the recipient entity.

If the 'Enable flag' is set to FALSE (0x00), the server cluster shall stop the information delivery service. However the device supporting this server cluster (i.e., Information node) shall still accept those commands needed to configure the node: Update, Delete, Configure Node Description, Configure Delivery Enable, Configure Push Information timer and Configure Set Root ID.

All cluster specific commands except from "configuration" commands will be replied by their respective responses with status code REQUEST_DENIED if the Delivery Enable attribute is disabled.

The Configure Delivery Enable command will be acknowledged with Default Response with status field equal to NOT_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e., Enable Secure Configuration attribute set to TRUE) while the sender didn't use the proper link key for sending the configuration commands.

Figure 12-22. Payload Format for Configure Delivery Enable Command

Octets:	**1**
Field:	Enable flag

12.2.4.3.9 Configure Push Information Timer

Payload format for the Configure Push Information Timer command shall be as illustrated in Figure 12-23.

Upon recipient of this command, the server side of the Information cluster shall change its Push Information timer attribute to the value of the 'Timer' field carried in this command.

The Configure Push Information Timer command will be acknowledged with Default Response with status field equal to NOT_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e., Enable Secure Configuration attribute set to TRUE) while the sender didn't use the proper link key for sending the configuration commands.

Figure 12-23. Payload Format for Configure Push Information Timer Command

Octets:	**4**
Field:	Timer

12.2.4.3.10 Configure Set Root ID

Payload format for the Configuration Set Root ID command shall be as illustrated in Figure 12-24.

Upon recipient of this command, the server side of Information cluster shall change its Root ID attribute to the value of the 'Root ID' field in this command.

The Configure Set Root ID command will be acknowledged with Default Response with status field equal to NOT_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e., Enable Secure Configuration attribute set to TRUE) while the sender didn't use the proper link key for sending the configuration commands.

Figure 12-24. Payload Format for Configure Set Root ID Command

Octets:	**2**
Field:	Root ID

12.2.4.4 Commands Generated

The generated command IDs for the Information cluster are listed in Table 12-11. Please notice that at least one of the following commands shall be implemented.

Table 12-11. Generated Command IDs for the Information Cluster

Command Identifier Field Value	**Description**	**M/O**	**Command Type**
0x00	Request Information Response	M	*Operation*
0x01	Push Information	M	*Operation*

Command Identifier Field Value	Description	M/O	Command Type
0x02	Send Preference Response	O	*Operation*
0x03	Server Request Preference	O	*Operation*
0x04	Request Preference Confirmation	O	*Operation*
0x05	Update Response	O	*Configuration*
0x06	Delete Response	O	*Configuration*

12.2.4.4.1 Request Information Response Command

This command is a response command according to a Request Information command which a client requests and carries requested information or carries status feedback if error occurs. Payload format for this command shall be as illustrated in Figure 12-25.

Figure 12-25. Payload Format of Request Information Response Command

Octet:	**1**	**1**	**Variable**	**…**	**1**	**Variable**
Field:	Number	Status Feedback 1	Single Content or ContentID	…	Status Feedback *n*	Single Content or ContentID

Number indicates how many single contents are carried by this command. Pairs of ‘Status Feedback’ and ‘Single Content’ appear corresponding to this number.

Single content is the actual content which format is specified in 12.2.6.1.

Status Feedback carries a message as a response to the previous ‘Request Information’ command sent by the client. Possible messages are SUCCESS, FAILURE, NOT_FOUND, MALFORMED_COMMAND, UNSUP_CLUSTER_COMMAND, INVALID_FIELD, HARDWARE_FAILURE and SOFTWARE_FAILURE already specified in enumeration list. Besides, INDICATION_REDIRECTION_TO_AP, REQUEST_DENIED, PREFERENCE_IGNORED and MULTIPLE_REQUEST_NOT_ALLOWED are included to these messages for this cluster specification. All the Status enumerations are listed in Table 12-12.

If a content is not available due to some reason (an error in many cases), the ‘Single Content’ field should be replaced by the “Content ID” in order to report which content requested has an error. (i.e., all the status codes except SUCCESS and PREFERENCE_IGNORED).

12.2.4.4.2 Push Information Command

This is a command putting information especially from an information node (or an access point) to a mobile terminal on push basis. A content sent to mobile terminal (e.g., a list of contents, a content described in octets strings, character strings, a title of content or RSS feed) is carried by this command.

Figure 12-26. Payload Format of Push Information Command

Octet:	**Variable**
Field:	Contents Data

Format for Contents Data shall be as illustrated in 12.2.6.

12.2.4.4.3 Send Preference Response Command

This command is used by the server to notify whether the data carried by Send Preference Command generated by the client is accepted correctly or not.

Payload format shall be as illustrated in Figure 12-27.

Status Feedback carries a message as a response to the previous command 'Send Preference' command from the client. Possible values are: SUCCESS, ZCL_PREFERENCE_DENIED, ZCL_PREFERENCE_IGNORED. If all the Preference Data are correctly processed, it is enough to respond with a unique Status Feedback equals to SUCCESS.

Also, if the server device does not support the Preference Type carried by the command, a unique Status Feedback value will be set to ZCL_PREFERENCE_IGNORED (0x74).

Figure 12-27. Payload Format for Send Preference Response Command and Request Preference Confirmation Command

Octet:	**1**	**…**	**1**
Field:	Status Feedback 1	…	Status Feedback *n*

12.2.4.4.4 Server Request Preference Command

This command requests a Preference as user-side information in the MT on pull-basis.

Upon receipt of this command at client cluster in the MT, the client is required to respond by Request Preference Response command.

This command has no payload.

12.2.4.4.5 Request Preference Confirmation Command

This command is used by the server to notify whether the data carried by Request Preference Response command generated by the client is accepted correctly or not.

Payload format shall be as illustrated in Figure 12-27 above.

Status Feedback carries a message as a response to the previous command 'Request Preference Response' command from the client. Possible values are: SUCCESS, ZCL_PREFERENCE_DENIED, ZCL_PREFERENCE_IGNORED. If all the Preference Data are correctly processed, it is enough to respond with a unique Status Feedback equals to SUCCESS.

Also, if the server device does not support the Preference Type carried by the command, a unique Status Feedback value will be set to ZCL_PREFERENCE_IGNORED (0x74).

12.2.4.4.6 Update Response Command

This command is used to notify any result of Update Command received by IN.

Payload format for this command shall be as illustrated in Figure 12-28.

Notification field has two sub-fields, content ID and Status Feedback. Content ID indicates what content the notification has the status for. Status Feedback indicates the status of the reception of the content.

Figure 12-28. Payload Format of Update Response and Delete Response command

Octet:	**2**	**1**	**…**	**2**	**1**
Field:	Notification 1		…	Notification *n*	
	Content ID 1	Status Feedback 1		Content ID *n*	Status Feedback *n*

Status Feedback carries a message as a response to the previous command 'Update' command from the client. Possible messages are SUCCESS, FAILURE, MALFORMED_COMMAND, UNSUP_CLUSTER_ COMMAND, INVALID_FIELD, INSUFFICIENT_SPACE, HARDWARE_FAILURE and SOFTWARE_FAILURE already specified in enumeration lists of ZCL [R1]. Besides, REQUEST_DENIED is included into these messages for this cluster specification. All the status enumerations are listed in Table 12-12.

12.2.4.4.7 Delete Response Command

This command is used to notify any result of Delete Command received by IN.

Payload format for this command shall be as illustrated in Figure 12-28.

Notification field has two sub-fields, content ID and Status Feedback. Content ID indicates what content the notification has the status for. Status Feedback indicates the status of the reception of the content.

12.2.5 Client

12.2.5.1 Command Received

The client receives the cluster specific commands detailed in 12.2.4.4.

12.2.5.2 Command Generated

The client generates the cluster specific commands detailed in 12.2.4.3, as required by application.

12.2.6 Payload Formats for Contents Data

This section describes about payload format for contents data as used in the commands defied for the Information cluster.

12.2.6.1 Payload Format for Multiple Contents

Payload format for the contents shall be as illustrated in Figure 12-29.

Figure 12-29. Payload Format for Multiple Contents

Octet	1	Variable	…	Variable
Data Type	uint8	Format for Single Content (defined in this section)		Format for Single Content (defined in this section)
Field Name	Number	Single Content 1	…	Single Content *n*

Number field holds a number of single contents. The payload format for the single content is specified in Figure 12-30.

Figure 12-30. Format for Single Content

Octet	2	1	Variable	Variable	1	2/0	…	2/0
Data Type	uint16	map8	Long Character String (defined in this cluster section)	Payload Format for 'Content' (defined in this cluster section)	uint8	uint16	….	uint16
Field Name	Content ID	Data Type ID	Title String	Content String	Number of children	Content ID 1	…	Content ID *n*

Content ID corresponds to the content. There is no rule provided by this document. It is expected to be defined by the service provider.

Data Type indicates the supported data types of content (it could be title and/or long octet, long character string or RSS). If a combination of type is supported by a Single Content, the order of data types shall be the one described inFigure 12-30. If a bit field of 'Title' in Data Type ID is 0b1, 'Title String' field will be inserted in the Single Content frame. If another bit than the 'Title' field is 0b1, 'content strings' field and following fields appear.

Title String appears in the Single Content frame only if a 'Title' bit in Data Type ID field is flagged. It represents title string in 'character string' data type; 'long character string' data type already includes 2 bytes count field.

Content String holds actual content data described in data type. It is inserted in the frame only if another bit than the 'Title' field is 0b1 in the Data Type ID field.

Number of Children indicates how many links to child-contents this content has. If there is no child for this content this field shall be set to 0.

Content ID n holds List of child-contents ID.

12.2.6.2 Contents Data Types

12.2.6.2.1 Title String

Figure 12-31. Format for Title String

Octet: 2	Variable
Count	Title

12.2.6.2.2 Long Octet String

Extended count field to two bytes. Count represents how many octets the Octet Data's length is.

Figure 12-32. Format for Long Octet String

Octet: 2	Variable
Count	Octet Data

12.2.6.2.3 Long Character String

Extended count field to two bytes. Count represents how many characters the Character Data's length is. It should not be in Bytes if the character set is not 8-bit code (e.g., 2-bytes code).

Figure 12-33. Format for Long Character String

Octet: 2	Variable
Count	Character Data

12.2.6.2.4 RSS Feed

Length field represents length in bytes not in character count. What character set is used should be defined in RSS feed data. In many cases, it would be ASCII compatible coding – like a UTF-8.

Figure 12-34. Format for RSS Feed

Octet: 2	Variable
Length	RSS Feed Data

12.2.6.3 Status Codes for the Information Cluster

Where an information cluster command contains a status field, the actual value of the enumerated status values are listed in Table 12-12. Because this table copied from ZCL status code enumeration and is inserted the Information cluster specific status codes at the start point, it may differ from the original if ZCL is updated. However, there is no problem because this table is used only for information cluster and is only used by its specific commands.

18029 **Table 12-12. Enumerated Status Values Used in the ZCL**

Enumerated status	**Value**	**Description**
SUCCESS	0x00	Operation was successful.
FAILURE	0x01	Operation was not successful.
-	0x02 – 0x6f	Reserved.
REQUEST_DENIED	0x70	Request was denied due to lack of permission
MULTIPLE_REQUEST_NOT_ALLOWED	0x71	Request of multiple contents is not supported
INDICATION_REDIRECTION_TO_AP	0x72	Server Indicates to change the access to the AP for this content.
PREFERENCE_DENIED	0x73	The preference was not accepted. Because it was not understandable or invalid.
PREFERENCE_IGNORED	0x74	Inform that preference was not used to modify the content which is provided by this command.
MALFORMED_COMMAND	0x80	The command appears to contain the wrong fields, as detected either by the presence of one or more invalid field entries or by there being missing fields. Command not carried out. Implementer has discretion as to whether to return this error or INVALID_FIELD.
UNSUP_CLUSTER_COMMAND	0x81	The specified general ZCL command is not supported on the device. Command not carried out.
INVALID_FIELD	0x85	At least one field of the command contains an incorrect value, according to the specification the device is implemented to. Command not carried out.
INSUFFICIENT_SPACE	0x89	An attempt to create an entry in a table failed due to an insufficient amount of free space available.
NOT_FOUND	0x8b	The requested information (e.g., table entry) could not be found.
-	0x8d – 0xbf	Reserved
HARDWARE_FAILURE	0xc0	An operation was unsuccessful due to a hardware failure.
SOFTWARE_FAILURE	0xc1	An operation was unsuccessful due to a software failure.
-	0xc3 – 0xff	Reserved

12.3 Chatting

12.3.1 Introduction

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

12.3.1.1 Scope and Purpose

This section specifies the Chatting cluster, which provides commands and attributes for sending chat messages among ZigBee devices. This cluster is to provide a standardized interface for people using ZigBee devices to chat with each other like they using instant messaging applications through Internet. The transaction sequence numbers used in the ZCL command frames for the Chatting cluster should be the same for the requests and responses; the default responses should use also the same transaction sequence numbers of the related commands in order to match the correspondent packets.

There are two kinds of chatting scenarios:

6. Centralized Server

   In this kind of scenario a centralized server is used for managing and controlling the messaging between the different ZigBee nodes. Different chat sessions can be made available by the server. The node entering the ZigBee network may search for the available chat sessions and join one of them after choosing one out of different available sessions. Different nodes can join one chat session and can interact with each other.

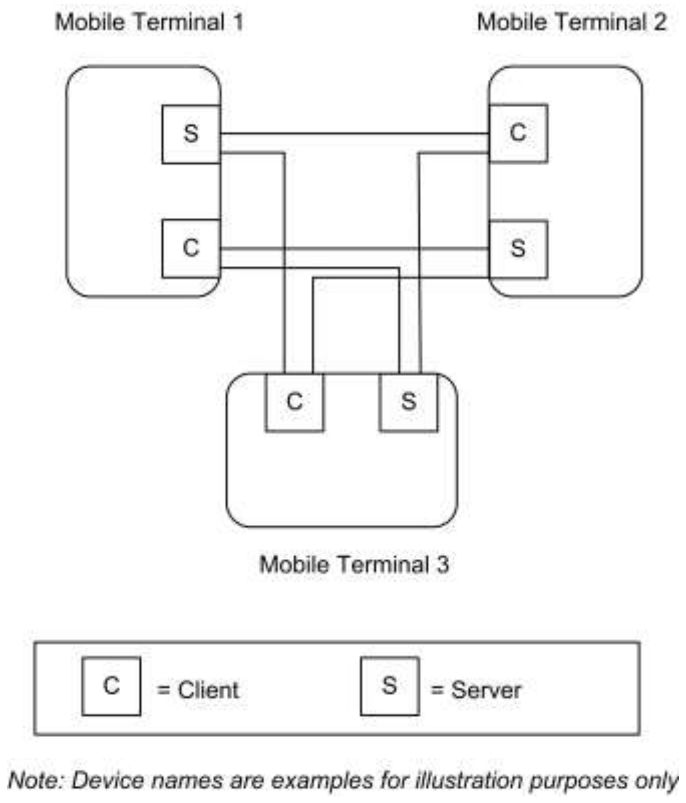
7. Ad Hoc Chat Sessions

   In this kind of scenario no infrastructure is needed. Any ZigBee node can start and manage a chat session. A ZigBee node in a particular ZigBee network can start a chat session. A node should only be a chairman of one chat session, i.e., it should only start one chat session. It is recommended to do so, since in the ad hoc scenario, the chairman can be any devices which may have low computing power and capability, and maintaining more than one session may be difficult for the devices. To start a chat session it has to decide a unique identifier for the chat session. This identifier shall be unique among all the chat sessions in the networks. For this requirement the implementer shall make it mandatory for a node to select a chat identifier which will be unique in the whole ZigBee network. The identifier may be set the same as the address of the device that starts the session, so as to guarantee its uniqueness.

This document should be used in conjunction with Chapter 2, Foundation, which gives an overview of the library and specifies the frame formats and general commands used therein.

This cluster provides attributes and commands for devices to send chatting messages to each other.

Figure 12-35. Typical Usage of the Chatting Cluster



12.3.1.2 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

12.3.1.3 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	CHAT	Type 1 (client to server)

12.3.1.4 Cluster Identifiers

Identifier	Name
0x0905	Chatting

12.3.2 Server

The server manages the list of participants, the chat session ID, etc. It can respond to the devices which are going to join chat sessions, or it can ask someone to leave the chat session. The server has the functions which are more related to managing the session than sending chatting messages.

The messages in the chat sessions are usually sent by the multicast method. So the server should also manage the chat group. There is an example which can be a guideline for implementing. Once the server forms a new chat session, it should form a new group. The group ID may be the same as the chat session ID. If a new user joins the chat session, it should also join the chat group. To fulfill this, the server should use the Add Group command specified in groups cluster to add the newcomer to the chat group. And if a user leaves the chat session, it should also leave the chat group. To fulfill this, the server should use the Remove Group command specified in groups cluster to remove the user from the chat group.

12.3.2.1 Dependencies

This cluster does not depend on any other existing clusters. However, in order to successfully fulfill the chatting, Information cluster, Groups cluster and Billing cluster may also need to be implemented in the same device where the chatting cluster is implemented.

12.3.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 12-13.

Table 12-13. Chatting Attributes Sets

Attribute Set Identifier	Description
0x000	User Related
0x001	Chat Session Related

12.3.2.2.1 User Related Attribute Set

The User Related Attribute Set contains the attributes summarized in Table 12-14.

Table 12-14. Attributes of the User Related Attribute Set

Identifier	Name	Type	Range	Access	M/O
0x0000	*U_ID*	uint16	0x0000-0xffff	R	M
0x0001	*Nickname*	string		R	M

12.3.2.2.1.1 U_ID Attribute

The *U_ID* attribute is the unique identification of the user in the chat room. It may be same as the address given to the device while joining in the ZigBee network, or may be same as the 2 least significant bytes of UserID of Billing cluster. The value 0xffff means this attribute is not set.

12.3.2.2.1.2 Nickname Attribute

The *Nickname* attribute is a unique display name of the user identified by the *U_ID* while talking in the public chat room. User sets the *Nickname* while joining the chat room.

12.3.2.2.2 Chat Session Related Attribute Set

The Chat Session Related set contains the attributes summarized in Table 12-15.

Table 12-15. Attributes of Chat Session Related Attribute Set

Identifier	Name	Type	Range	Access	M/O
0x0010	*C_ID*	uint16	0x0000-0xffff	R	M
0x0011	*Name*	string		R	M
0x0012	*EnableAddChat*	bool	TRUE/FALSE	R	O

12.3.2.2.2.1 C_ID Attribute

The *C_ID* attribute is the unique identification of a chat room. It is assigned by the chat server while creating a new chat room following the user command or chosen by the chairman. It may be same as the chat group ID. If the server maintains several chat rooms, this attribute should be set as the ID of the latest formed chat room. The value 0xffff means this attribute is not set.

12.3.2.2.2.2 Name Attribute

The *Name* attribute is the name or topic of the chat room which is identified by the *C_ID* attribute.

12.3.2.2.2.3 EnableAddChat Attribute

The *EnableAddChat* attribute indicates whether the server permits other users to add new chat rooms in it. TRUE (0x01) indicates the server permit other users to add new chat rooms, while FALSE (0x00) indicates not permit to do so.

12.3.2.3 Commands Received

The received commands IDs for the chatting cluster are listed in Table 12-16.

Table 12-16. Command IDs for the Chatting Cluster

Command Identifier Field Value	Description	M/O
0x00	Join Chat Request	M
0x01	Leave Chat Request	M
0x02	Search Chat Request	M
0x03	Switch Chairman Response	O
0x04	Start Chat Request	O
0x05	ChatMessage	M
0x06	Get Node Information Request	O

12.3.2.3.1 Join Chat Request Command

The Join Chat Request command is used for a node to request to join one chatting session.

The Join Chat request command shall be formatted as illustrated in Figure 12-36.

Figure 12-36. Format of the Join Chat Request Command

Octets	2	Variable	2
Data Type	uint16	string	uint16
Field Name	U_ID	Nickname	C_ID

The U_ID field indicates unique identification of the user in the chat room.

The Nickname field is type of character string which is a unique display name of the user while talking in the public chat room.

The C_ID field is unique identification of a chat room. It indicates the ID of the chat room which the client wants to join.

This command should be unicast to the server which manages the chat room indicated by the C_ID.

12.3.2.3.2 Leave Chat Request Command

The Leave Chat Request command is used for a node to request to leave one chatting session. The client may require the Default Response command to be sent from the server so that to confirm the leave command has been successfully received.

The Leave Chat request command shall be formatted as illustrated in Figure 12-37.

Figure 12-37. Format of the Leave Chat Request Command

Octets	2	2
Data Type	uint16	uint16
Field Name	C_ID	U_ID

The C_ID field indicates the ID of the chat room which the node wants to leave. The U_ID field indicates unique identification of the user in the chat room.

This command should be unicast to the server which manages the chat room the user to leave.

12.3.2.3.3 Search Chat Request Command

The Search Chat Request command is used for a node to request to search for the available chat session on the server.

The Search Chat Request command shall contain no payload and shall be originated by the devices which want to have a chat with others and sent to the server. It may be broadcast in the network.

12.3.2.3.4 Switch Chairman Response Command

The Switch Chairman Response command is used for nodes to response to the Switch Chairman Request command.

The Switch Chairman Response command shall be formatted as illustrated in Figure 12-38.

Figure 12-38. Format of the Switch Chairman Response Command

Octets	2	2
Data Type	uint16	uint16
Field Name	C_ID	U_ID

The C_ID field in the command indicates the ID of the chat room of which the receiving node is the old chairman. The U_ID field in the command is the unique ID of node which wants to be the chairman of the chat room.

This command shall be unicast to the chairman, announcing that the node indicated by the U_ID volunteers to be the new chairman.

12.3.2.3.5 Start Chat Request Command

The Start Chat Request command is used for a device to request to create one chat session. The new chat session to be created shall be managed by the responder. That is, once the chat session is created, the responder but not the requester will be the chairman of the chat room.

The Start Chat request command shall be formatted as illustrated in Figure 12-39.

Figure 12-39. Format of the Start Chat Request Command

Octets	Variable	2	Variable
Data Type	string	uint16	string
Field Name	Name	U_ID	Nickname

The Name field indicates the topic of the chat room. The U_ID field indicates unique identification of the user in the chat room. The Nickname field indicates the Nickname set by the requester.

The command is originated by the devices which want to create one chat room and attract others who have the same interest in the topic. It should be unicast to the server which manages the chat rooms.

12.3.2.3.6 ChatMessage Command

The *ChatMessage* command is used for chatting, i.e., one node to send a message to other nodes. In the case of peer chatting, such as to exchange some private messages, it may be unicast to the chairman first, the chairman may forward this command with unicast method to the destination user. The *ChatMessage* command may be sent directly to the destination node in peer chatting case if the destination network address and endpoint are known by the sender. *Get Node Information Request* and *Response* commands shall be used to acquire the necessary network address and endpoint information. In the case of normal chatting (a message to be sent to the whole room), all nodes in the same chat room are expected to receive the message. The ChatMessage command should be multicast to other nodes in the chat room.

In peer chatting case, if the command contains illegal parameter such as non-existing U_ID field, the chairman should return a Default Response command with INVALID_FIELD status.

The *ChatMessage* command shall be formatted as illustrated in Figure 12-40.

Figure 12-40. Format of the ChatMessage Command

Octets	2	2	2	Variable	Variable
Data Type	uint16	uint16	uint16	string	string
Field Name	Destination U_ID	Source U_ID	C_ID	Nickname	Message

The Destination U_ID field indicates the destination node’s U_ID. The Source U_ID field indicates the source node’s U_ID. The C_ID indicates the ID of the chat room which the sender belongs to. The Nickname field indicates the sender’s Nickname, which shall be in Character string data type.

In the case of peer chatting, the Destination U_ID field and the Source U_ID field shall be set to the specific nodes' U_ID. In the case of normal chatting (sending a message to all the users in the chat room), Destination U_ID shall be set to 0xffff while Source U_ID shall be set to the specific source node's U_ID.

12.3.2.3.7 Get Node Information Request Command

The Get Node Information Request command is used for peer chatting to get the network address and endpoint number of the peer node, so as to use ChatMessage command to send private message to the node. When one wants to send private massage to another node in the same chatting session, it shall check whether it has that node’s network address and endpoint number. If not, it shall send this command to the server.

The Get Node Information Request command shall be formatted as illustrated inFigure 12-41.

Figure 12-41. Format of the Get Node Information Request Command

Octets	2	2
Data Type	uint16	uint16
Field Name	C_ID	U_ID

The C_ID field indicates the ID of the chat room which the investigated node belongs to. The U_ID field indicates the U_ID of the node to be investigated.

This command should be unicast to the chairman node. A chatting table should be maintained by the chairman. It may be also maintained by other nodes. When a chairman has assigned a U_ID to a node it shall add related information into the chatting table, and when a node leaves the chatting session it shall remove the record of the leaving device from the table. A node may get the address of another node from the chairman by using the Get Node Information Request command. Once it gets the information, the node may store it for future usage. The detail format of the chatting table is implementer dependent. An example of each item of the table may be illustrated as Figure 12-42. The node number field indicates the number of NodeInformation field. The NodeInformation field is as specified in Figure 12-50.

Figure 12-42. Format of an Item of the Chatting Table

C_ID	Node number	NodeInformation 1	…	NodeInformation *n*

12.3.2.4 Commands Generated

The generated commands IDs for the Chatting cluster are listed in Table 12-17.

Table 12-17. Generated Command IDs for the Chatting Cluster

Command Identifier Field Value	Description	M/O
0x00	Start Chat Response	O
0x01	Join Chat Response	M
0x02	User Left	M
0x03	User Joined	M
0x04	Search Chat Response	M
0x05	Switch Chairman Request	O
0x06	Switch Chairman Confirm	O
0x07	Switch Chairman Notification	O
0x08	Get Node Information Response	O

12.3.2.4.1 Start Chat Response Command

The Start Chat Response command is used for server to response to the Start Chat request command. If successful, the server shall then form a new chat room and make itself the chairman.

The Start Chat Response command shall be formatted as illustrated in Figure 12-43.

Figure 12-43. Format of the Start Chat Response Command

Octets	1	0/2
Data Type	enum8	uint16
Field Name	Status	C_ID

The Status field indicates the status of the previous request. . If it is SUCCESS, the C_ID field shall exist, or else the C_ID field shall not exist. The C_ID field indicates the unique identification of the chat room and it is assigned by the server. If the server doesn't permit to add a new chat room, i.e., the attribute EnableAddChat being set to FALSE, the server shall return this command with FAILURE Status.

This command shall be unicast to the requester.

12.3.2.4.2 Join Chat Response Command

The Join Chat Response is used for server to response the Join Chat Request command.

The Join Chat Response shall be formatted as illustrated in Figure 12-44.

Figure 12-44. Format of the Join Chat Response Command

Octets	1	2	0/2	Variable	Variable	0/2	Variable
Data Type	enum8	uint16	uint16	string	-	uint16	string
Field Name	Status	C_ID	U_ID 1	Nickname 1	…	U_ID *n*	Nickname *n*

The Status field indicates the status of the previous request.. If it is SUCCESS, the list of the U_ID and Nickname fields shall exist, or else the list shall not exist. The C_ID field indicates the ID of the chat room which the server manages. It shall be the same as the C_ID field in the corresponding Join Chat Request command. The list of the U_ID and Nickname fields indicate other participants in the chat room. Each U_ID field and the Nickname field respectively indicate the unique ID and the nickname of each user in the chat room.

This command shall be unicast to the requester. After receiving this command, the node should check whether it has received the Add Group command from the chairman. If not, it should wait for that command so as to know which group it belongs to. How long it should wait for the command is specific to the implementation.

12.3.2.4.3 User Left Command

The User Left command is used for server to inform other participants that someone has left the chat room.

The User Left shall be formatted as illustrated in Figure 12-45.

Figure 12-45. Format of the User Left Command

Octets	2	2	Variable
Data Type	uint16	uint16	string
Field Name	C_ID	U_ID	Nickname

The C_ID indicates the ID of the chat room which the user left. The U_ID field indicates the left participant's unique ID in the chat room. The Nickname field is the nickname of the left participant.

The command shall be multicast to all users in the same chat room.

12.3.2.4.4 User Joined Command

The User Joined command is used for server to inform other participants that someone has just joined the chat room.

The User Joined command shall be formatted as illustrated in Figure 12-46.

Figure 12-46. Format of the User Joined Command

Octets	2	2	Variable
Data Type	uint16	uint16	string
Field Name	C_ID	U_ID	Nickname

The C_ID indicates the ID of the chat room which the newcomer joined. The U_ID field indicates the newcomer's unique ID in the chat room. The Nickname field is the nickname of the newcomer.

This command should be multicast to all users in the same chat room. When the newcomer receives the command, it shall compare the U_ID field in the command with U_ID of itself, if same, it shall ignore the command.

12.3.2.4.5 Search Chat Response Command

The Search Chat Response command is used for server to respond to the Search Chat Request command.

The Search Chat Response command shall be formatted as illustrated in Figure 12-47.

Figure 12-47. Format of the Search Chat Response command

Octets	1	0/2	Variable	Variable	0/2	Variable
Data Type	map8	uint16	string	…	uint16	string
Field Name	Options	C_ID 1	Name 1	…	C_ID *n*	Name *n*

The Options field indicates the options of this command. Bit 0 of the Options field indicates whether the server permits other users to add new chat rooms in it. The value 0b0 means permit while 0b1 means not permit. Other bits of this field are reserved. The list of the C_ID and Name fields indicates the information of the available chat rooms. Each C_ID field and Name field indicate respectively the chat room identification and topic of each available chat room. It's recommended at least one chat room should be maintained by the server. If no chat room is maintained, the list of C_ID and Name fields shall not exist.

This command should be unicast to the requester. Only the chairman can send out this command, and the list of chat room information shall only contain the information of the chat rooms which it manages. The server may also broadcast this command to notify other users which chat rooms it manages. After receiving this command, the network address and endpoint number should be extracted from the network layer header and APS header, so as to acquiring the chairman's network address and endpoint number.

12.3.2.4.6 Switch Chairman Request Command

In the case of Ad-Hoc chat, when a chairman wants to leave the chat session, he can use this command to appoint a new chairman out of the participating Devices which can continue to manage the chat room.

The Switch Chairman Request command shall be multicast to every device which is in the same chat room. It shall be formatted as illustrated in Figure 12-48.

Figure 12-48. Format of the Switch Chairman Request Command

Octets	2
Data Type	uint16
Field Name	C_ID

The C_ID field indicates the ID of the chat room where the chairman is requested to be changed.

12.3.2.4.7 Switch Chairman Confirm Command

The Switch Chairman Confirm command is used by the old chairman to inform the node which the chairman has selected to be the new chairman.

The Switch Chairman Confirm command shall be formatted as illustrated in Figure 12-49.

Figure 12-49. Format of the Switch Chairman Confirm Command

Octets	2	Variable	Variable	Variable
Data Type	uint16	-	…	-
Field Name	C_ID	NodeInformation 1	…	NodeInformation *n*

The C_ID field indicates the ID of the chat room which the chairman manages. The NodeInformation field is formatted as illustrated in Figure 12-50. Each *NodeInformation* field contains information about a node participating in the chat session. This field shall contain the following sub-fields, the U_ID sub-field, *Address* sub-field, *Endpoint* sub-field and the *Nickname* sub-field. The U_ID sub-field, *Address* sub-field, *Endpoint* sub-field and *Nickname* sub-field indicate the node's unique ID, network address, endpoint number and nickname respectively. This command shall be unicast to the new chairman.

Figure 12-50. Format of the *NodeInformation* Field

Octets	2	2	1	Variable
Data Type	uint16	data16	uint8	string
Sub-field Name	U_ID	Address	Endpoint	Nickname

12.3.2.4.8 Switch Chairman Notification Command

The Switch Chairman Notification command is used by the old chairman to inform other participants in the chat room about the change in the chairman. The Switch Chairman Confirm command shall be formatted as illustrated in Figure 12-51.

Figure 12-51. Format of the Switch Chairman Notification Command

Octets	2	2	2	1
Data Type	uint16	uint16	data16	uint8
Field Name	C_ID	U_ID	Address	Endpoint

The C_ID field is the ID of the chat room which the chairman manages. The U_ID field is the unique ID of the node which is the new chairman of the chat room. The *Address* field and the *Endpoint* field are the network address and the endpoint number of the new chairman respectively. The command should be multicast to other nodes in the same chat room.

12.3.2.4.9 Get Node Information Response Command

The Get Node Information Response command is used by the server to give a response to the Get Node Information Request command, so that the requesting node can obtain the desired information including network address and endpoint number of a specific node. If successful, the server shall provide the node information in the response command.

The Get Node Information Response command shall be formatted as illustrated in Figure 12-52.

Figure 12-52. Format of the Get Node Information Response Command

Octets	1	2	2	0/2	0/1	Variable
Data Type	enum8	uint16	uint16	data16	uint8	string
Field Name	Status	C_ID	U_ID	Address	Endpoint	Nickname

The *Status* indicates the status of the previous request. If it is SUCCESS, the *Address* field, the *Endpoint* field and the *Nickname* field shall exist, or else those fields shall not exist. The C_ID field and the U_ID field shall be the same as the corresponding *Get Node Information Request* command. The C_ID field is the ID of the chat room which the investigated node belongs to. The U_ID field is the unique ID of the investigated node. The *Address* field, the *Endpoint* field and the *Nickname* field indicate the network address, the endpoint number and the nickname of the investigated node respectively. The command shall be unicast to requester. After receiving this command, the node may store the information of the investigated node for future usage.

12.3.3 Client

12.3.3.1 Commands Received

The client receives the cluster specific commands detailed in 12.3.2.4 as required by application profiles.

12.3.3.2 Commands Generated

The client generates the cluster specific commands detailed in 12.3.2.3 as required by application profiles.

12.4 Voice Over ZigBee

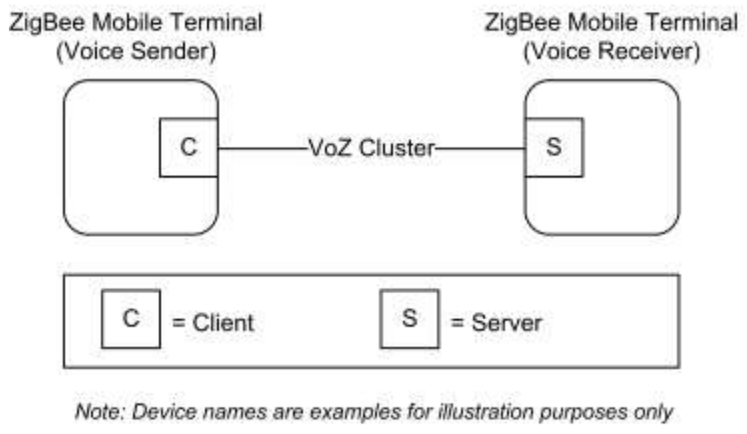
12.4.1 Scope and Purpose

This section specifies a single cluster, the VoZ cluster, which provides commands and attributes for voice receiving and transmitting among ZigBee devices. This cluster is to provide a standardized interface for the devices to receive/transmit voice data packets.

This section should be used in conjunction with Chapter 2, Foundation, which gives an overview of the library and specifies the frame formats and general commands used therein.

The cluster specified in this document is typically used for telecom applications, but may be used in any other application domains. This cluster may use Partition cluster.

Figure 12-53. Typical Usage of the VoZ Cluster


ZigBee Mobile Terminal (Voice Sender)
ZigBee Mobile Terminal (Voice Receiver)
C
VoZ Cluster
S
C = Client
S = Server
Note: Device names are examples for illustration purposes only

12.4.2 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides attributes and commands for devices to receive/transmit their voice data. One of the devices plays a role of a receiver and the other does that of a sender. For example, a receiver receives voice data from the other MT (voice sender).

An important thing to notice for this VoZ cluster is that there are two different types of service for this cluster. One of them is voice transmission between humans (human-to-human). The other type of the service is voice transmission from human to device (human-to-device voice data transmission, i.e., voice command) in order to send a voice 'command' to a device. Therefore, the meaning of 'voice' delivery includes not only human voice delivery (human-to-human), but also voice delivery for device control (human-to-device, i.e., voice command). These two types of service will be referenced whenever necessary.

12.4.2.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

12.4.2.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	VOZ	Type 1 (client to server)

12.4.2.3 Cluster Identifiers

Identifier	Name
0x0904	Voice Over ZigBee

12.4.3 Server

The server stores the data to be shared. It may response to the request from other devices and transmit the data to them, or it may actively request other devices to transmit the data to them.

12.4.3.1 Dependencies

None

12.4.3.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 12-18.

Table 12-18. VoZ Attribute Sets

Attribute Set Identifier	Description
0x000	Voice Information
0x001 – 0xfff	Reserved

12.4.3.2.1 Establishment Information Attribute Set

The Establishment Information attribute set contains the attributes summarized in Table 12-19.

Table 12-19. Attributes of the Voice Information Attribute Set

Id	Name	Type	Range	Access	M/O
0x0000	*CodecType*	enum8	G.711(PCM) =0x01 G.726(ADPCM)=0x02 CELP =0x03 AMR =0x04	RW	M
0x0001	*SamplingFrequency*	enum8	SF_8K =0x01 SF_7K =0x02 SF_3_5K =0x03	RW	M

Id	Name	Type	Range	Access	M/O
0x0002	*Codecrate*	enum8	CR_64K =0x01 CR_40K =0x02 CR_32K =0x03 CR_24K =0x04 CR_16K =0x05 CR_8K =0x06 CR_6_3K =0x07 CR_5_3K =0x08 CR_AMR-NB =0x09 CR_AMR-WB=0x0a	RW	M
0x0003	*Establishment Timeout*	uint8	0x01-0xff	-	M
0x0004	*CodecTypeSub1*	enum8	-	RW	O
0x0005	*CodecTypeSub2*	enum8	-	RW	O
0x0006	*CodecTypeSub3*	enum8	-	RW	O
0x0007	*CompressionType*	enum8	ALaw =0x01 uLaw =0x02	-	O
0x0008	*CompressionRate*	enum8	-	-	O
0x0009	*OptionFlags*	map8	0x00-0xff	RW	O
0x000a	*Threshold*	uint8	0x00-0xff	RW	O

12.4.3.2.1.1 CodecType Attribute

The *CodecType* attribute specifies the enumeration of the codec type. G.711 Codec is PCM (pulse code modulation) method by the ITU-T. G.726 Codec is ADPCM (adaptive differential PCM) method which has involved G.721 and G.723 ITU-T codec. CELP (code excited linear prediction) is voice codec of CDMA-based digital mobile communication system. AMR (adaptive multirate codec) is used to 3GP European mobile equipment.

12.4.3.2.1.2 SamplingFrequency Attribute

The *SamplingFrequency* attribute specifies the enumeration of the sampling frequency (Hz).

PCM, ADPCM, CELP: 8KHz, AMR: 3.5KHz, 7KHz

12.4.3.2.1.3 CodecRate Attribute

The *CodecRate* attribute specifies the enumeration of the codec rate (Kbps).

Various codec rates available.

PCM: 64Kbps, ADPCM: 40/32/24/16Kbps, CELP: 5.3/8Kbps, AMR: 5 ~ 12Kbps

12.4.3.2.1.4 EstablishmentTimeout Attribute

The *EstablishmentTimeout* attribute sets timeout value to 1/10 sec in order to disconnect an establishment between devices when there is no response after the establishment.

12.4.3.2.1.5 CodecTypeSub1, CodecTypeSub2, CodecTypeSub3 Attribute

CodecTypeSub1, *CodecTypeSub2*, and *CodecTypeSub3* attributes are used for additionally supportable Codecs other than the system default one. It has the same range value as that of CodecType attribute.

12.4.3.2.1.6 CompressionType Attribute

The *CompressionType* attribute specifies the enumeration of the compression type

ALaw: the compression technology for transmission data to minimize the quantification error in PCM, (Europe)

uLaw: the compression technology for transmission data to minimize the quantification error in PCM, (US, Japan)

12.4.3.2.1.7 CompressionRate Attribute

The *CodecRate* attribute specifies the enumeration of compression rate.

Compression rate is defined based on compression type.

12.4.3.2.1.8 OptionFlags Attribute

The *OptionFlags* attribute indicates the optional function. It shall be formatted as illustrated in Figure 12-54.

Figure 12-54. Format of the OptionFlags Attribute

Bits: b0	b1	b2	b3-b7
Occupancy	PLC	VAD	Reserved

The Occupancy field specifies whether the occupancy sensor is active or not. If the Occupancy field is set to one, it indicates the occupancy sensor is active. If the Occupancy field is set to zero, it indicates the occupancy sensor is inactive.

PLC (Packet Loss Concealment): enabled in logic level high in order to correct voice data when there is loss for the data

VAD (Voice Activity Detection): enabled in logic level high in order to distinguish mute voice data from non-mute voice data

12.4.3.2.1.9 Threshold Attribute

The *Threshold* attribute specifies the value for voice loudness in voice transmission.

12.4.3.3 Commands Received

The received command IDs for the VoZ cluster are listed in Table 12-20.

Before proceeding, please refer to the section 12.4.2 of overview, and especially, to the service type that there are two types of service in this cluster. The commands in this section are developed and used not only for human-to-human voice delivery, but also for human-to-device voice delivery in order to send a voice command to control the device.

Table 12-20. Command IDs for the VoZ Cluster

Command Identifier Field Value	Description	M/O
0x00	Establishment Request	M
0x01	Voice Transmission	M
0x02	Voice Transmission Completion	O
0x03	Control Response	O

12.4.3.3.1 Establishment Request Command

The Establishment Request command is used for a device to request for a connection of the voice information from another device. It shall be originated by the voice transmission source device and sent to the transmission destination device.

The establishment request command shall be formatted as illustrated in Figure 12-55.

For Codec Type, Sampling Frequency, Codec Rate and etc., please refer to Table 12-19. For Service Type, please refer to section 12.4.2; the Service Type equal to 0x00 indicates human-human service and 0x01 indicates human-device service. Most commands in this section are developed and used for human-to-device communication.

Figure 12-55. Format of the Establishment Request Command

Octets	Data Type	Field Name
1	map8	Flag
1	enum8	Codec Type
1	enum8	Samp. Freq.
1	enum8	Codec Rate
1	enum8	Service Type
1/0	enum8	Codec TypeS1
1/0	enum8	Codec TypeS2
1/0	enum8	Codec TypeS3
1/0	enum8	Comp. Type
1/0	enum8	Comp. Rate

The Flag field value of Figure 12-55 is set according to the bit values of Figure 12-56 when a VoZ device has an optional attribute value such as CodecTypeSub1.

Figure 12-56. Format of the Flag

Bits: b0	b1	b2	b3	b4-b7
CodecTypeSub1	CodecTypeSub2	CodecTypeSub3	Compression	Reserved

12.4.3.3.2 Voice Transmission Command

The Voice Transmission command is used for a device to transmit the voice data to other devices. It shall be originated by the voice transmission source device and sent to the voice transmission destination device. If required, Partition Cluster should be used for this command.

In case of transmitting multiple voice data, the Sequence Number in the ZCL Header should be sequentially increased in order to detect the loss of data and reassemble them.

Figure 12-57. Format of the Voice Transmission Command

Octets	Variable
Data Type	-
Field Name	Voice Data

12.4.3.3.3 Voice Transmission Completion

The Voice Transmission Completion command is sent to the destination device when needed, after the source device transmits all voice data which should be transmitted.

The voice transmission command completion shall be formatted as illustrated in Figure 12-58.

Figure 12-58. Format of the Voice Transmission Completion Command

Octets	Variable
Data Type	-
Field Name	ZCL Header

12.4.3.3.4 Control Response Command

The Control Response command is used to respond with the success or failure of the control, when a device receives the Control command.

The voice control response command shall be formatted as illustrated in Figure 12-59.

Figure 12-59. Format of the Control Response Command

Octets	1
Data Type	enum8
Field Name	ACK=0x01 NAK=0x00

12.4.3.4 Commands Generated

The generated command IDs for the VoZ cluster are listed in Table 12-21.

Before proceeding, please refer to the section 12.4.2 of overview, and especially, to the service type that there are two types of service in this cluster. The commands in this section are developed and used not only for human-to-human voice delivery, but also for human-to-device voice delivery in order to send a voice command to control the device.

Table 12-21. Generated Command IDs for the VoZ Cluster

Command Identifier Field Value	Description	M/O
0x00	Establishment Response	M
0x01	Voice Transmission Response	M
0x02	Control	O

12.4.3.4.1 Voice Transmission Response Command

The Voice Transmission Response command is to notify the sender of NACK. It shall be originated by the voice transmission destination device and sent to the voice transmission source device.

The voice transmission response command shall be formatted as illustrated in Figure 12-60.

Figure 12-60. Format of the Voice Transmission Response Command

Octets	1	1
Data Type	uint8	enum8
Field Name	Sequence Number of ZCL Header	Error Flag

If there is an error in processing the received voice data, the receiving device should respond with the Voice Transmission Response command with the sequence number of ZCL Header and the Error Flag set to an error reason according to Table 12-22.

Table 12-22. The Error Flag of Voice Transmission Response

Error Flag Identifier Field Value	Description
0x00	Failure to decode voice data
0x01	Wrong order of voice data

12.4.3.4.2 Establishment Response Command

The Establishment Response command is to notify the device which previously requests for connecting the voice information. It shall be originated by the voice transmission destination device and sent to the voice transmission source device.

The Establishment Response command shall be formatted as illustrated in Figure 12-61.

Figure 12-61. Format of the Establishment Response Command

Octets	1	1/0
Data Type	enum8	enum8
Field Name	ACK=0x01 NAK=0x00	CodecType

When a receiving device receives the Establishment Request command with *CodecType* which is not supported, it responds with the Establishment Response command with NAK and *CodecType* supported by the device.

If the requested *CodecType* exists among *CodecTypeSub1*, *CodecTypeSub2*, and *CodecTypeSub3*, the *CodecType* field is set to the value.

If the device receives the Establishment Response command with NAK and *CodecType*, it first checks whether it supports the *CodecType* in the received command. If it supports, the device transmits the Establishment Request command with its *CodecType* again.

12.4.3.4.3 Control Command

The Control command is to control the voice transmission source device. It shall be originated by the voice transmission destination device and sent to the voice transmission source device.

For example, this command is used for such as walkie-talkie communication or radio listening.

The voice control command shall be formatted as illustrated in Figure 12-62.

Figure 12-62. Format of the Control Command

Octets	1
Data Type	enum8
Field Name	Control Type

The Control Type field indicates the control options, including the play operation (0x01), the stop operation (0x02), and the disconnection operation (0x03). The play operation is to request for starting voice data transmission. The stop operation is to request for stopping voice data transmission. The disconnection operation is to terminate the connection between the voice transmission source/destination devices.

12.4.4 Client

12.4.4.1 Command Received

The client receives the cluster specific commands detailed in 12.4.3.4 as required by application profiles

12.4.4.2 Command Generated

The client generates the cluster specific commands detailed in 12.4.3.3 as required by application profiles.

CHAPTER 13 COMMISSIONING

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

13.1 General Description

13.1.1 13.1.1 Introduction

This chapter contains commissioning methods for devices that can be used in any application domain.

13.1.2 13.1.2 Cluster List

This section lists the clusters specified in this document. The clusters defined in this document are listed in Table 13-1.

Table 13-1. Clusters for Commissioning

ID	Cluster Name	Description
0x0015	Commissioning	The commands and attributes for commissioning a device onto the network
0x1000	Touchlink	The commands and attributes for Touchlink commissioning a device

13.2 Commissioning

13.2.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides attributes and commands pertaining to the commissioning and management of devices operating in a network.

This cluster will typically be supported using a “Commissioning Tool.” But, depending on the application and installation scenario, this tool may take many forms. For purposes of this document, any device that implements the client side of this cluster may be considered a commissioning tool.

As with all clusters defined in the Cluster Library, an application may have as many instances of this cluster as needed and may place them on any addressable endpoint.

This cluster is exclusively used for commissioning the ZigBee stack and defining device behavior with respect to the ZigBee network. It does not apply to applications operating on those devices.

13.2.1.1 Security and Authorization

The attributes and commands covered in this cluster specification are critical to the operation of a ZigBee device. An application entity that receives a request to access the attributes of this cluster or to execute one of the commands described in sub-clause 13.2.2.3 shall determine whether the originator is authorized to make that request and whether the security processing applied to the received frame was appropriate. The method or methods whereby this is accomplished are out of the scope of this document but it is strongly recommended that Entity Authentication, as described in [B1], be used. This, and any other methods used to authorize commissioning tools and other devices acting as a client for this cluster, shall be detailed in any Application Profile documents that use it.

Similarly, it is strongly recommended that the cluster specified here be deployed only on a single device endpoint or that, at very least, all deployments of this cluster be managed by a single application object with a unitary set of security requirements etc.

13.2.1.2 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

13.2.1.3 Classification

Hierarchy	Role	PICS Code
Base	Utility	CS

13.2.1.4 Cluster Identifiers

Identifier	Name
0x0015	Commissioning

13.2.2 Server

The attributes accessible on the server side of this cluster are typically attributes of the ZigBee stack, which are either described in the layer Information Base for some stack layer, or are ZDO configuration attributes. The function of the server is to provide read/write access to these attributes and to manage changes of certain critical attributes in a way that prevents the device from getting into an inconsistent and unrecoverable state.

Thus, for example, the application entity that receives and processes commands to set attributes in the Startup Parameters attribute set shall check whether the *StartupControl* attribute has been set to a value that is inconsistent with the value of the *ExtendedPanID* attribute (see Table 13-2). If such a condition arises, e.g., a request is made to set the *StartupControl* attribute to 0x02, indicating network rejoin, and simultaneously to clear the *ExtendedPanID* attribute indicating an unspecified network, then an error (INCONSISTENT_STARTUP_STATE) shall be reported.

13.2.2.1 Dependencies

None

13.2.2.2 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 13-1.

Table 13-1. Commissioning Attribute Sets

Attribute Set Identifier	Description
0x000, 0x001	Startup Parameters
0x002	Join Parameters
0x003	End Device Parameters
0x004	Concentrator Parameters

For each of these sets, each attribute is mandatory unless specifically specified as optional in the relevant sub-clause defining it. Similarly, any default values are specified in these sub-clauses.

13.2.2.2.1 Startup Parameters Attribute Set

The Startup Parameters attribute set contains the attributes summarized in Table 13-2.

These are application attributes and, as such, are sent, received and managed by application entities. However, except where otherwise noted, each of them corresponds to, and is intended to provide a value for a particular stack attribute that controls the startup behavior of the stack. The ZigBee specification describes a schematic startup procedure (see [B1]), which governs the order and manner in which these stack attributes must be used in order to gain access to a network or form a new network. This procedure should run when a device starts up, but may also run without an actual restart as part of the ongoing operation of the device.

The Restart Device command (see 13.2.2.3.1) provides a means whereby a set of Startup Parameters - the “current” Startup Parameters attribute set - stored at the application layer, can be installed in the stack and put into force by executing the startup procedure described above and in the specification. A change to one of the attributes contained in this set, e.g., the *ShortAddress* attribute, does not immediately result in a change to the underlying stack attribute. The attribute set will be installed on receipt of a Restart Device command.

Note that the attributes in this set are mutually interdependent and must be taken as a whole. One consequence of this is that, while there are no explicit requirements with regard to storage class for these attributes, implementers must carefully consider whether to make a particular attribute non-volatile or static in order to prevent inconsistencies in the attribute set after an unintentional processor restart. Another consequence is that, wherever possible, startup attributes should be written atomically using a single write attributes command frame.

Table 13-2. Attributes of the Startup Parameters Attribute Set

Id	Name	Type	Range	Def	Acc	MO
0x0000	ShortAddress	uint16	0x0000 – 0xfff7	-	RW	M
0x0001	ExtendedPANId	EUI64	0x0000000000000000 - 0xfffffffffffffffe	0xffffffffffffffff	RW	M

Id	Name	Type	Range	Def	Acc	MO
0x0002	PANId	uint16	0x0000 - 0xffff	-	RW	M
0x0003	Channelmask	map32	Any valid IEEE 802.15.4 channel mask (see [E1]).	-	RW	M
0x0004	ProtocolVersion	uint8	0x02	-	RW	M
0x0005	StackProfile	uint8	0x01 - 0x02	-	RW	M
0x0006	StartupControl	enum8	0x00 - 0x03	-	RW	M
0x0010	TrustCenterAddress	EUI64	Any valid IEEE Address	*all zeros*	RW	M
0x0011	TrustCenterMasterKey	key128	Any 128-bit value	*all zeros*	RW	O
0x0012	NetworkKey	key128	Any 128-bit value	*all zeros*	RW	M
0x0013	UseInsecureJoin	bool	FALSE/TRUE	TRUE	RW	M
0x0014	PreconfiguredLinkKey	key128	Any 128-bit value	*all zeros*	RW	M
0x0015	NetworkKeySeqNum	uint8	0x00 - 0xff	0x00	RW	M
0x0016	NetworkKeyType	enum8	Any valid key type value	-	RW	M
0x0017	NetworkManagerAddress	uint16	Any valid network address	0x000	RW	M

Except where specifically noted, an implementer of this cluster shall provide read access to all attributes of the Startup Parameters attribute set. However, if an attempt is made to read an attribute that may not be read, a WRITE_ONLY status value shall be returned.

Even in cases where the commissioning cluster is a mandatory part of a given application profile, an implementer is not required to provide write access for all attributes. If write access is not provided, it is assumed that the implementer has some other preferred, generally out-of-band, method for setting the value of the underlying stack attribute, and that the value returned on read reflects the actual value in use. If an attempt is made to write to such an attribute, a DEFINED_OUT_OF_BAND status value shall be returned

13.2.2.2.1.1 ShortAddress Attribute

The *ShortAddress* attribute contains the intended 16-bit network address of the device. This attribute corresponds to the *nwkShortAddress* attribute of the NIB (see [B1]).

The default value is the value stored in the *nwkShortAddress* attribute of the NIB. When this attribute is not set as part of the Restart Device Request command, this default value ensures that the previous short address is preserved. This makes it possible for a device to preserve its short address after being commissioned.

Stack profile compatibility for this attribute is described in Table 13-3.

Table 13-3. Stack Profile Compatibility for the *ShortAddress* Attribute

StackProfile Value	Supported	Comment
0x01	No	Under the ZigBee stack profile a ZigBee router or device shall obtain a network address from its parent at network formation time.
0x02	Yes	Under the ZigBee PRO stack profile and stochastic addressing a device may, under certain circumstances, generate its own network address and keep it through the joining process (see [B1]). In this case, it may make sense for that address to be provided by a tool if, for example, this will reduce the likelihood of address conflicts.

13.2.2.2.1.2 ExtendedPANId Attribute

The *ExtendedPANId* attribute holds the extended PAN Id of the network of which the device should be a member. See 13.2.2.2.1.7 for usage details.

The default value of 0xffffffffffffffff indicates an unspecified value. In the case where a device is required to join a commissioning network on startup, this attribute may be set, under application control, to the global commissioning EPID (see 13.2.4.1).

Depending in the value of the *StartupControl* attribute, this attribute may correspond to the *nwkExtendedPANID* attribute of the NIB (see [B1]) or the *apsUseExtendedPANID* attribute of the AIB (see [B1]).

13.2.2.2.1.3 *PANId* Attribute

The *PANId* attribute holds the PAN Id of the network of which the device should be a member. This attribute corresponds to the *macPANId* attribute of the MAC PIB (see [E1]).

The default value is macPANID.

Stack profile compatibility for this attribute is described in Table 13-4.

Table 13-4. Stack Profile Compatibility for the *PANId* Attribute

StackProfile Value	Comment
0x01	Under the ZigBee stack profile, The ZigBee coordinator shall select an appropriate PANId at network formation time. In this case the value of the PANId attribute may be used. A ZigBee router or ZigBee end device shall obtain a PANId from its parent at network join time. In this case, the value of the PANId attribute shall be ignored.
0x02	Under the ZigBee PRO stack profile a ZigBee router or end device that has the StartupControl attribute equal to 0x00, must have the PANId attribute set to the correct value since it has no other way of obtaining it.

13.2.2.2.1.4 ChannelMask Attribute

The *ChannelMask* attribute is an IEEE802.15.4 channel mask, see [E1], containing the set of channels the device should scan as part of the network join or formation procedures. This attribute corresponds to the *apsChannelMask* attribute of the AIB (see [B1]).

The default value is the value of apsChannelMask.

13.2.2.2.1.5 ProtocolVersion Attribute

The *ProtocolVersion* attribute is used to select the current protocol version for a device that supports multiple versions of the ZigBee specification.

This attribute is optional. A device may support a single protocol version or multiple protocol versions at the option of the implementer.

Currently only one value, 0x02 denoting ZigBee 2006 and later, is supported. The default value shall be the protocol version supported by the application if only one protocol version is supported. Should more than one protocol version be supported, the default value may be any of the protocol versions supported.

The *ProtocolVersion* attribute corresponds to a NWK layer constant, *nwkcProtocolVersion*, which is defined as a constant because most implementations will support only a single ZigBee protocol version. In this case, the attribute will be read-only. However, there is nothing to prevent a device with sufficient resources from supporting more than one ZigBee protocol version under control of the commissioning cluster.

13.2.2.2.1.6 StackProfile Attribute

The *StackProfile* attribute is used to select the stack profile for the device.

This attribute is optional. A device may only support one stack profile.

Supported values include:

0x01: ZigBee Stack profile

0x02: ZigBee PRO Stack Profile

This attribute corresponds to the *nwkStackProfile* attribute of the NIB (see [B1]). The default value shall be the stack profile supported by the application if only one stack profile is supported. Should more than one stack profile be supported, the default value may be any of the stack profiles supported.

13.2.2.2.1.7 StartupControl Attribute

The *StartupControl* attribute is an enumerated type that determines how certain other parameters are to be used. Values for this attribute and interaction with other attributes are shown in Table 13-5. If an attribute appears in the “required attributes” column this indicates that this attribute must be set to a value that is valid for the intended operational network in order for this StartupControl attribute value to be used. Note that in some cases the default value may be sufficient.

If an attribute appears in the “optional attributes” column it means that the attribute value will affect startup or operation under the given attribute set but that any value, including the default, is a valid value. If an attribute appears in the “ignored attributes” column it means that the value of this attribute has no effect on device startup when the StartupControl attribute value in the “value” column is in force.

18610 **Table 13-5. *StartupControl* Attribute Usage**

Value	Description	Required Attributes	Optional Attributes	Ignored Attributes
0x00	Indicates that the device should consider itself part of the network indicated by the *ExtendedPANId* attribute. In this case it will not perform any explicit join or rejoin operation.	ShortAddress, ExtendedPANId, PANId, TrustCenterAddress, NetworkKey, NetworkKeySeqNum, NetworkKeyType	ChannelMask, UseInsecureJoin, NetworkManagerAddress, TrustCenterMasterKey (required for Stack Profile 2, optional for Stack Profile 1), PreconfiguredLinkKey	-
0x01	Indicates that the device should form a network with extended PAN ID given by the *ExtendedPANId* attribute. The AIB attribute *apsDesignatedCoordinator* (see [B1]) shall be set to TRUE in this case.	ExtendedPANId	PANId, ChannelMask, NetworkManagerAddress, NetworkKey, NetworkKeyType, TrustCenterAddress	ShortAddress, UseInsecureJoin NetworkKeySeqNum, TrustCenterMasterKey, PreconfiguredLinkKey
0x02	Indicates that the device should rejoin the network with extended PAN ID given by the *ExtendedPANId* attribute. The AIB attribute *apsDesignatedCoordinator* (see [B1]) shall be set to FALSE in this case.	ExtendedPANId	ShortAddress, ChannelMask, UseInsecureJoin, NetworkKey, NetworkKeySeqNum, NetworkKeyType TrustCenterAddress, TrustCenterMasterKey, NetworkManagerAddress, PreconfiguredLinkKey	PANId
0x03	Indicates that the device should start "from scratch" and join the network using (unsecured) MAC association. The AIB attribute *apsDesignatedCoordinator* (see [B1]) shall be set to FALSE in this case.	-	ExtendedPANId, ChannelMask, PreconfiguredLinkKey	ShortAddress, UseInsecureJoin, PANId, TrustCenterAddress, NetworkKey, NetworkKeySeqNum, NetworkKeyType, NetworkManagerAddress, TrustCenterMasterKey

Note that these values control the execution of the device startup procedure as specified in [B1], sub-clause 2.5.5.5.6.2. See this sub-clause for a detailed description of the operation of this procedure.

The default value of the StartupControl attribute for an un-commissioned device is 0x03.

Stack profile compatibility for this attribute is shown in Table 13-6.

Table 13-6. Stack Profile Compatibility for the *StartupControl* Attribute

StackProfile Value	StartupControl Value		Comment
	Mandatory	Optional	
0x01	0x01 0x03	0x02	ZigBee networks use tree-structured address assignment and must form, at startup, from the ZigBee coordinator. The "mode" implied by *StartupControl* = 0 in which a device is essentially preconfigured to run on a network without having to explicitly join in order to get an address or PAN Id is not supported.
0x02	0x01 0x03	0x00 0x02	StartupControl = 0 is supported under the ZigBee Pro stack profile.

Note: An implementation shall return an error code of INVALID_VALUE when a client attempts to write an unsupported *StartupContol* value.

13.2.2.2.1.8 TrustCenterAddress Attribute

The trust center address to use when performing security operations on the network whose extended PAN ID is given by the *ExtendedPANId* attribute is, in turn, given by the *TrustCenterAddress* attribute.

This attribute corresponds to the *apsTrustCenterAddress* attribute of the AIB (see [B1].

The default value of 0x0000000000000000 indicates unspecified.

13.2.2.2.1.9 TrustCenterMasterKey Attribute

This attribute holds the trust center master key to use during key establishment with the TC of the network with the extended PAN ID given by the ExtendedPANId attribute.

The default value, i.e., a 128-bit value containing all zeros, indicates that the key is unspecified.

This attribute corresponds to the MasterKey element of the key-pair set from the *apsDeviceKeyPairSet* attribute of the AIB for which the DeviceAddress element corresponds to the value of the *TrustCenterAddress* attribute. (see [B1]).

13.2.2.2.1.10 *NetworkKey* Attribute

This attribute supplies the NWK key to use when communicating with the network specified by the *ExtendedPANId* attribute. The default value, i.e., a 128-bit value containing all zeros, indicates that the key is unspecified.

This attribute corresponds to the active key from the *nwkSecurityMaterialSet* attribute of the NIB (see [B1].

13.2.2.2.1.11 UseInsecureJoin Attribute

This attribute is a Boolean flag that enables the use of unsecured join as a fallback case at startup time. It corresponds to the Boolean AIB attribute *apsUseInsecureJoin* (see [B1]. The default value is TRUE.

13.2.2.2.1.12 PreconfiguredLinkKey Attribute

The preconfigured link key is the key between the device and the trust center. The default value, i.e., a 128-bit value containing all zeros, indicates that the key is unspecified.

This attribute corresponds to the LinkKey element of the Key-Pair descriptor contained in the *apsDeviceKeyPairSet* attribute of the AIB (see [B1]).

13.2.2.2.1.13 NetworkKeySeqNum Attribute

This attribute sets the network key's sequence number. The default value is 0x00.

This attribute corresponds to the value of the *nwkActiveKeySeqNumber* attribute of the NIB (see [B1]).

13.2.2.2.1.14 NetworkKeyType Attribute

This attribute sets the network key's type. It corresponds to the value of the KeyType element of the current security material descriptor corresponding to the Trust Center found in the nwkSecurityMaterialSet attribute of the NIB (see [B1]).

The default value is 0x01 when the StackProfile is 0x01 and 0x05 when the StackProfile is 0x02.

13.2.2.2.1.15 NetworkManagerAddress Attribute

This attribute sets the address of the Network Manager. It corresponds to the value of the *nwkManagerAddr* attribute of the NIB (see [B1]).

The default value is 0x0000 indicating that, by default, the Network Manager is on the ZigBee coordinator.

13.2.2.2.2 Join Parameters Attribute Set

The Join Parameters attribute set contains the attributes summarized in Table 13-7.

These attributes control the details of the network joining process. Each of them, as described below, corresponds to a ZDO configuration attribute, the function and use of which is described in the ZigBee specification (see [B1]).

Table 13-7. Attributes of the Join Parameters Attribute Set

Id	Name	Type	Range	Def	Acc	M/O
0x0020	ScanAttempts	uint8	0x001 – 0xff	0x05	RW	O
0x0021	TimeBetweenScans	uint16	0x0001 - 0xffff	0x64	RW	O
0x0022	RejoinInterval	uint16	0x0001 - *MaxRejoinInterval*	0x3c	RW	O
0x0023	MaxRejoinInterval	uint16	0x0001 - 0xffff	0x0e10	RW	O

As with the attributes in Table 13-2, an implementer of this cluster shall provide read access to all attributes. The implementer may provide write access. If write access is not provided, it is assumed that the implementer has some other preferred method for setting the value of the underlying stack attribute, and that the value returned on read reflects the actual value in use.

13.2.2.2.2.1 ScanAttempts Attribute

The *ScanAttempts* attribute determines how many scan attempts to make before selecting the ZigBee Coordinator or Router to join.

This attribute corresponds to the *:Config_NWK_Scan_Attempts* configuration attribute of the ZDO (see [B1]).

The default value for this attribute is 0x05.

13.2.2.2.2.2 TimeBetweenScans Attribute

The *TimeBetweenScans* attribute determines the time between each scan attempt.

This attribute corresponds to the *:Config_NWK_Time_btwn_Scans* configuration attribute of the ZDO (see [B1]).

The units of this attribute are milliseconds and the default value is 0x64.

13.2.2.2.2.3 RejoinInterval Attribute

The *RejoinInterval* determines the interval between attempts to rejoin the network if an end device finds itself disconnected.

This attribute corresponds to the *:Config_Rejoin_Interval* configuration attribute of the ZDO (see [B1]).

The units of this attribute are seconds and the default value is 0x3c.

13.2.2.2.2.4 MaxRejoinInterval Attribute

The *MaxRejoinInterval* attribute imposes an upper bound on the RejoinInterval parameter.

This attribute corresponds to the *:Config_Max_Rejoin_Interval* configuration attribute of the ZDO (see [B1]).

The units of this attribute are seconds and the default value is 0x0e10.

13.2.2.2.3 End Device Parameters Attribute Set

The End Device Parameters attribute set contains the attributes summarized in Table 13-8.

Table 13-8. Attributes of the End Device Parameters Attribute Set

Id	Name	Type	Range	Def	Acc	M/O
0x0030	IndirectPollRate	uint16	0x0000 – 0xffff	-	RW	O
0x0031	ParentRetryThreshold	uint8	0x00 - 0xff	-	R	O

As with the attributes in Table 13-2 and Table 13-7, an implementer of this cluster shall provide read access to all attributes. The implementer may provide write access. If write access is not provided, it is assumed that the implementer has some other preferred method for setting the value of the underlying stack attribute, and that the value returned on read reflects the actual value in use.

13.2.2.2.3.1 IndirectPollRate Attribute

The *IndirectPollRate* attribute determines the rate at which a device, usually an end device, where the *macRxOnWhenIdle* attribute of the PIB has a value of FALSE, will poll for messages from its parent.

This attribute corresponds to the *:Config_NWK_IndirectPollRate* configuration attribute of the ZDO (see [B1]).

The units for this attribute are milliseconds and the default value, broad limits for which are given in [Z2] and [Z3], shall be determined by the relevant application. Values assigned using this cluster should be within the given limits in order to promote correct network operation.

13.2.2.2.3.2 ParentRetryThreshold Attribute

The *ParentRetryThreshold* attribute determines how many times a ZigBee end device should attempt to contact its parent before initiating the rejoin process. ZigBee routers and ZigBee coordinators should return a value of 0xff for this attribute on read, and should return an error on any attempt to write it.

This attribute corresponds to the *:Config_Parent_Link_Retry_Threshold* configuration attribute of the ZDO (see [B1]).

13.2.2.2.4 Concentrator Parameters Attribute Set

The Concentrator Parameters attribute set contains the attributes summarized in Table 13-9.

Table 13-9. Attributes of the Concentrator Parameters Attribute Set

Ide	Name	Type	Range	Def	Acc	M/O
0x0040	ConcentratorFlag	bool	FALSE/TRUE	FALSE	RW	O
0x0041	ConcentratorRadius	uint8	0x00 - 0xff	0x0f	RW	O
0x0042	ConcentratorDiscoveryTime	uint8	0x00 - 0xff	0x0000	RW	O

As with the other attribute sets in this cluster, an implementer shall provide read access to all attributes. The implementer may provide write access. If write access is not provided, it is assumed that the implementer has some other preferred method for setting the value of the underlying stack attribute, and that the value returned on read reflects the actual value in use.

13.2.2.2.4.1 *ConcentratorFlag* Attribute

The *ConcentratorFlag* attribute will configure the device to be a concentrator for the purpose of many-to-one routing. This attribute corresponds to the *nwkIsConcentrator* attribute of the NIB (see [B1]).

The default value for this attribute is FALSE.

13.2.2.2.4.2 ConcentratorRadius Attribute

The *ConcentratorRadius* attribute determines the hop count radius for concentrator route discoveries. This attribute corresponds to the *nwkConcentratorRadius* attribute of the NIB (see [B1]).

The default value for this attribute is 0x0f.

13.2.2.2.4.3 ConcentratorDiscoveryTime Attribute

Routes to the concentrator are known as inbound routes. These routes are created after the receipt of a command from the concentrator. The *ConcentratorDiscoveryTime* attribute determines the period for triggering such route creation.

This attribute corresponds to the *nwkConcentratorDiscoveryTime* attribute of the NIB (see [B1]).

The units of this attribute are seconds and the default value is 0x0000, which indicates that the discovery time is unknown and must be performed by the application.

13.2.2.3 Commands Received

The received command IDs for the commissioning cluster server are listed in Table 13-10. These commands may, in principle, be received as unicasts or as broadcasts, but application developers should be aware that, since these commands require a response, broadcasting them to a large number of devices may not be advisable.

Table 13-10. Commands Received by the Commissioning Cluster Server

Command Identifier	Description	M/O
0x00	Restart Device	M
0x01	Save Startup Parameters	O
0x02	Restore Startup Parameters	O
0x03	Reset Startup Parameters	M

In Table 13-10, if the actions associated with an optional command are not implemented, at least the relevant response command (see Table 13-12) must be returned with status UNSUP_CLUSTER_COMMAND.

13.2.2.3.1 Restart Device Command

The Restart Device command is used to optionally install a set of startup parameters in a device and run the startup procedure so as to put the new values into effect. The new values may take effect immediately or after an optional delay with optional jitter. The server will send a Restart Device Response command back to the client device before executing the procedure or starting the countdown timer required to time the delay.

13.2.2.3.1.1 Payload Format

The Restart Device command is formatted as shown in Figure 13-1.

Figure 13-1. Format of the Restart Device Command Payload

Octets	1	1	1
Data Type	8-bit bitmap	Unsigned 8-bit integer	Unsigned 8-bit integer
Field Name	Options	Delay	Jitter

Figure 13-2. Format of the Options Field

Bits: 0...2	3	4...7
Startup Mode	Immediate	Reserved

The Startup Mode sub-field of the options field is 3 bits in length and shall take one of the nonreserved values from Table 13-11.

Table 13-11. Startup Mode Sub-field Values

Field value	Description
0b000	Restart the device using, i.e., installing, the current set of startup parameters.
0b001	Restart the device using, and not replacing, the current state of the device, i.e., the current set of stack attributes.

The Immediate sub-field of the options field is 1 bit in length. If this sub-field has a value of 1 then the device is to execute the restart either immediately on receipt of the Restart Device Request frame, if the value of the delay field is 0, or immediately after the prescribed delay and jitter has transpired if not. If the immediate sub-field has a value of 0, then the device may wait to restart until after the prescribed delay and jitter, if any, have transpired but may also wait for a "convenient" moment, e.g., until pending frames have been transmitted, to actually perform the restart.

The delay field is one octet in length and gives a delay in seconds, in the range [0…255], after which the startup procedure is to be invoked.

The jitter field is one octet in length and specifies a random jitter range. While possible field values fall in the interval [0…255], the actual jitter, in milliseconds, that should be added to the delay, given in seconds, in the delay field should be:

RAND(<jitter field contents> * 80) ms.

Where RAND(X) returns a random number in the interval [0…X].

13.2.2.3.1.2 Effect on Receipt

On receipt of the Restart Device command, the application checks the current startup attribute set for consistency. If the attribute set is incorrect or inconsistent, processing of the command is terminated and a Restart Device Response command is returned to the sender of the request with a status value of INCONSISTENT_STARTUP_STATE. Otherwise, the application sends a Restart Device Response command to the sender of the request with a status value of SUCCESS, then leaves the current network, installs the current startup attribute set, if the startup mode sub-field of the options field has a value of 0b00, and runs the restart procedure after the given delay and jitter have transpired.

13.2.2.3.2 Save Startup Parameters Command

In addition to the current set of startup parameters, which every device implementing the commissioning cluster must maintain, a device may store and maintain up to 256 sets of startup attributes. The Save Startup Parameters Request command allows for the current attribute set to be stored under a given index. Note that while the startup attribute set index is 8 bits, allowing for as many as 256 attribute sets, the actual number of attribute sets will typically be much smaller.

While storage of additional startup attribute sets is optional, a device that chooses to store additional startup attribute sets must store them in such a way that they are non-volatile.

13.2.2.3.2.1 Payload Format

The Save Startup Parameters command is formatted as shown in Figure 13-3.

Figure 13-3. Format of Save Startup Parameters Command Payload

Octets	1	1
Data Type	8-bit bitmap	Unsigned 8-bit integer
Field Name	Options (Reserved)	Index

The Options field is one octet in length and is reserved.

The Index field is one octet in length and gives an index under which the current startup parameter attribute set is to be saved.

13.2.2.3.2.2 Effect on Receipt

On receipt of the Save Startup Parameters command, the application shall check the value of the index field of the command payload. If the index field has a value that is equal to an index under which a set of startup parameters has already been saved then the current startup parameters attribute set is simply saved in place of the previously saved set and a Save Startup Parameters Response command is sent back to the sender of the request with a status value of SUCCESS.

If the value of the index field is such that no startup parameters attribute set has been saved under that index then the application shall check that there is storage capacity to save another attribute set. If there is capacity then the current startup parameters attribute set shall be stored under the index given in the index field such that it may be restored at a future time in response to the receipt of a Restore Startup Parameters Request command carrying the same index. A Save Startup Parameters Response command with status value of SUCCESS is then sent as described above.

If there is not storage capacity, then a save Startup Parameters Response command is sent back to the sender of the request with a status INSUFFICIENT_SPACE.

13.2.2.3.3 Restore Startup Parameters Command

A device that implements the optional Save Startup Parameters command shall also implement the Restore Startup Parameters Request command (and vice-versa). This command allows a saved startup parameters attribute set to be restored to current status overwriting whatever was there previously.

13.2.2.3.3.1 Payload Format

The Restore Startup Parameters command is formatted as shown in Figure 13-4.

Figure 13-4. Restore Startup Parameters Command Payload

Octets	1	1
Data Type	8-bit bitmap	Unsigned 8-bit integer
Field Name	Options (Reserved)	Index

The options field is one octet in length and is reserved.

The index field is one octet in length and gives the index of the saved startup parameter attribute set to be restored to current status.

13.2.2.3.3.2 Effect on Receipt

On receipt of the Restore Startup Parameters command, the application shall check the value of the index field of the command payload. If the index field has a value that is equal to an index under which a startup parameters attribute set has been saved then that attribute set is copied into the current startup parameters attribute set overwriting whatever was there and a Restore Startup Parameters Response command is sent back to the sender of the request with a status value of SUCCESS. If the value of the index field is such that no startup parameters attribute set has been saved under that index then a Restore Startup Parameters Response command is sent back to the sender of the request with a status value of INVALID_FIELD.

13.2.2.3.4 Reset Startup Parameters Command

This command allows current startup parameters attribute set and one or all of the saved attribute sets to be set to default values. There is also an option for erasing the index under which an attribute set is saved thereby freeing up storage capacity.

13.2.2.3.4.1 Payload Format

The Reset Startup Parameters command is formatted as shown in Figure 13-5.

Figure 13-5. Format of Reset Startup Parameters Command Payload

Octets	1	1
Data Type	8-bit bitmap	Unsigned 8-bit integer
Field Name	Options	Index

The Options field is formatted as shown in Figure 13-6.

Figure 13-6. Format of the Options Field

Bits: 0	**1**	**2**	**3...7**
Reset Current	Reset All	Erase Index	Reserved

The Reset Current sub-field of the options field is 1 bit in length. If it has a value of 1 then all attributes in the current startup parameters attribute set shall be reset to their default values. Otherwise the current startup parameters attribute set shall remain unchanged.

The Reset All sub-field of the options field is 1 bit in length. If it has a value of 1 then all attributes of all saved startup parameter attribute sets shall be reset to their default values. Otherwise, all attributes of the saved attribute set with an index given by the value of the index field shall be set to their default values

The Erase Index sub-field of the options field is 1 bit in length. If it has a value of 1 then the index under which a saved attribute set has been saved shall be cleared as well, essentially freeing the storage associated with that index.

The Index field is one octet in length and gives the index of a saved startup parameter attribute set. The value of this field is ignored if either the reset all sub-field or the reset current sub-field of the options field have a value of 1.

13.2.2.3.4.2 Effect on Receipt

On receipt of the Reset Startup Parameters Request command the application interprets the options field and index field as described in sub-clause 13.2.2.3.4.1 and acts accordingly. The Reset Startup Parameters Response command sent back to the sender of the request shall always have a status value of SUCCESS.

13.2.2.4 Commands Generated

The command IDs for the commands generated by the commissioning cluster server are listed in Table 13-12.

Table 13-12. Commands Generated by the Commissioning Cluster Server

Command Identifier	Description	M/O
0x00	Restart Device Response	M
0x01	Save Startup Parameters Response	M
0x02	Restore Startup Parameters Response	M
0x03	Reset Startup Parameters Response	M

These commands should always be issued as unicasts.

13.2.2.4.1 Payload Format

All response commands emitted by the server have the same payload format as shown in Figure 13-7.

Figure 13-7. Format of Reset Startup Parameters Command Payload

Octets	1
Data Type	8-bit enumeration
Field Name	Status

Status values are chosen from the set of non-reserved values shown in Chapter 2

13.2.2.4.2 Effect on Receipt

On receipt of one of the response commands shown in Table 13-12, the client is made aware that the server has received the corresponding request and is informed of the status of the request.

13.2.3 Client

The commissioning cluster client (e.g., implemented on a commissioning tool) manages the attributes described above on a remote device and sends the Restart Device command as necessary.

13.2.3.1 Dependencies

None

13.2.3.2 Attributes

The client cluster has no attributes.

13.2.3.3 Commands Received

The client receives the cluster specific commands generated by the server (see 13.2.2.4).

13.2.3.4 Commands Generated

The client generates the cluster specific commands received by the server, as required by the application. See 13.2.2.3.

13.2.4 Commissioning EUI-64s

To assist in ensuring that commissioning can be achieved in an interoperable environment while minimizing the possibility of interference from existing or future ZigBee and 802.15.4 networks and devices a range of IEEE-defined 64-bit extended unique identifiers (EUI-64s), as been reserved for use as Extended PAN IDs. The reserved range is as follows:

- 00-50-C2-77-10-00-00-00 is the global commissioning EPID
- 00-50-C2-77-10-00-00-01 to 00-50-C2-77-10-00-FF-FF are EUI-64s reserved for other commissioning use

13.2.4.1 Global Commissioning EPID

The global commissioning EPID is intended to serve as a single EUI-64 to be used by any ZigBee application for the purpose of commissioning. It is recommended that profile and application developers that require interoperability between products offered by different OEMs incorporate this global commissioning EPID within their respective application profiles as the EPID that devices attempt to join when they are first turned on straight “out-of-the-box.”

This global commissioning EPID provides a guarantee that devices will join a specific network for commissioning purposes. As part of commissioning, devices are then provided with a startup attribute set (SAS) that ensures that they join a network other than this global commissioning network. These SASs may be provided over-the-air using the commissioning cluster or some other out-of-band method.

It is also recommended that this global commissioning EPID be used only for commissioning, and especially not for ongoing operational use. Commissioning networks formed using the global commissioning EPID should be temporary and such networks should be stopped upon completion of commissioning to minimize the possibility of such networks interfering with other attempts at forming commissioning networks.

13.2.4.2 EUI-64s Reserved for Other Uses

Additional EUI-64s have been reserved for other use by the Alliance. At this point, their intended usage has not been specified. These identifiers should not be used without prior agreement with the ZigBee Alliance. It is recommended that if a profile or application developer requires the use of these additional EUI-64s, they should contact the Core Stack Group (CSG) within the ZigBee Alliance.

13.3 Touchlink Commissioning

The *Touchlink 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 touchlink 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.

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 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.

13.3.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

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

13.3.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added
2	added Profile Interop bit in Scan Request frame, CCB 2115 2105

13.3.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	TL

13.3.1.3 Cluster Identifiers

Identifier	Name
0x1000	Touchlink Commissioning

13.3.2 Server

13.3.2.1 Attributes

The server has no attributes.

13.3.2.2 Commands Received

When a device implements the *touchlink commissioning* cluster at the ZCL server side, it shall be able to receive the commands listed in Table 13-13.

Table 13-13. Commands Received by the Server Side of the Touchlink Commissioning Cluster

	Command Identifier Field Value	Description	M/O	Reference
Touchlink	0x00	Scan request	M	13.3.2.2.1
	0x02	Device information request	M	13.3.2.2.2
	0x06	Identify request	M	13.3.2.2.3
	0x07	Reset to factory new request	M	13.3.2.2.4
	0x10	Network start request	M	13.3.2.2.5
	0x12	Network join router request	M	13.3.2.2.6
	0x14	Network join end device request	M	13.3.2.2.7
	0x16	Network update request	M	13.3.2.2.8
	All other values in the range 0x00 – 0x3f	*Reserved*	-	-
Utility	0x41	Get group identifiers request	O*	13.3.2.2.9
	0x42	Get endpoint list request	O*	13.3.2.2.10
	All other values in the range 0x40 – 0xff	*Reserved*	-	-

* These are mandatory for a controller device as defined in the device specification.

13.3.2.2.1 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 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 13-8.

Figure 13-8. Format of the Scan Request Command Frame

Octets	4	1	1
Data Type	Unsigned 32-bit integer	8-bit bitmap	8-bit bitmap
Field Name	Inter-PAN transaction identifier	ZigBee information	Touchlink information

13.3.2.2.1.1 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.

13.3.2.2.1.2 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 13-9.

Figure 13-9. Format of the ZigBee Information Field

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

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.

13.3.2.2.1.3 Touchlink information field

The *Touchlink information* field is 8-bits in length and specifies touchlink-specific information. This field shall be formatted as illustrated in Figure 13-10.

Figure 13-10. Format of the Scan Request Touchlink Information Field

Bits: 0	1	2-3	4	5	6	7
Factory new	Address assignment	Reserved	Link initiator	Undefined (can be 0 or 1)	Reserved	Profile Interop[156]

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 touchlink commissioning cluster at the client side) or 0 otherwise (i.e., it does not support the touchlink commissioning cluster at the client side).

The Profile Interop subfield is 1 bit in length and specifies which profile the device implements. If the ZLL profile is implemented, this bit shall be set to 0. In all other case (Profile Interop / ZigBee 3.0), this bit shall be set to 1.[157]

13.3.2.2.2 Device Information Request Command Frame

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

[156] CCB 2115
[157] CCB 2115

This inter-PAN command shall be formatted according to the general inter-PAN frame format 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 13-11.

Figure 13-11. Format of the Device Information Request Command Frame

Octets	4	1
Data Type	Unsigned 32-bit integer	Unsigned 8-bit integer
Field Name	Inter-PAN transaction identifier	Start index

13.3.2.2.2.1 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.

13.3.2.2.2.2 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.

13.3.2.2.3 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 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 13-12.

Figure 13-12. Format of the Identify Request Command Frame

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

13.3.2.2.3.1 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.

13.3.2.2.3.2 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 13-14.

Table 13-14. 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.

13.3.2.2.4 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 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 13-13.

Figure 13-13. Format of the Reset to Factory New Request Command Frame

Octets	4
Data Type	Unsigned 32-bit integer
Field Name	Inter-PAN transaction identifier

13.3.2.2.4.1 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.

13.3.2.2.5 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 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 13-14.

Figure 13-14. Format of the Network Start Request Command Frame

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

13.3.2.2.5.1 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.

13.3.2.2.5.2 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.

13.3.2.2.5.3 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.

13.3.2.2.5.4 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, encrypted according to the algorithm indicated by the *key index* field.

13.3.2.2.5.5 Logical Channel Field

The *logical channel* field is 8-bits in length and shall contain the touchlink 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.

13.3.2.2.5.6 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.

13.3.2.2.5.7 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.

13.3.2.2.5.8 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.

13.3.2.2.5.9 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.

13.3.2.2.5.10 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.

13.3.2.2.5.11 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.

13.3.2.2.5.12 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.

13.3.2.2.5.13 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.

13.3.2.2.5.14 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.

13.3.2.2.5.15 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.

13.3.2.2.6 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 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 13-15.

Figure 13-15. Format of the Network Join Router Request Command Frame

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

13.3.2.2.6.1 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.

13.3.2.2.6.2 Extended PAN Identifier Field

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

13.3.2.2.6.3 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.

13.3.2.2.6.4 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, encrypted according to the algorithm indicated by the *key index* field.

13.3.2.2.6.5 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.

13.3.2.2.6.6 Logical Channel Field

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

13.3.2.2.6.7 PAN Identifier Field

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

13.3.2.2.6.8 Network Address Field

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

13.3.2.2.6.9 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.

13.3.2.2.6.10 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.

13.3.2.2.6.11 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.

13.3.2.2.6.12 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.

13.3.2.2.6.13 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.

13.3.2.2.6.14 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.

13.3.2.2.7 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 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 13-16.

Figure 13-16. Format of the Network Join End Device Request Command Frame

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

13.3.2.2.7.1 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.

13.3.2.2.7.2 Extended PAN Identifier Field

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

13.3.2.2.7.3 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.

13.3.2.2.7.4 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, encrypted according to the algorithm indicated by the *key index* field.

13.3.2.2.7.5 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.

13.3.2.2.7.6 Logical Channel Field

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

13.3.2.2.7.7 PAN Identifier Field

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

13.3.2.2.7.8 Network Address Field

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

13.3.2.2.7.9 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.

13.3.2.2.7.10 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.

13.3.2.2.7.11 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.

13.3.2.2.7.12 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.

13.3.2.2.7.13 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.

13.3.2.2.7.14 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.

13.3.2.2.8 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 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 in Figure 13-17.

Figure 13-17. Format of the Network Update Request Command Frame

Octets	4	8	1	1	2	2
Data Type	uint32	IEEE address	uint8	uint8	uint16	uint16
Field Name	Inter-PAN transaction identifier	Extended PAN identifier	Network update identifier	Logical channel	PAN identifier	Network address

13.3.2.2.8.1 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.

13.3.2.2.8.2 Extended PAN Identifier Field

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

13.3.2.2.8.3 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.

13.3.2.2.8.4 Logical Channel Field

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

13.3.2.2.8.5 PAN Identifier Field

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

13.3.2.2.8.6 Network Address Field

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

13.3.2.2.9 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 13-18.

Figure 13-18. Format of the Get Group Identifiers Request Command

Octets	1
Data Type	Unsigned 8-bit integer
Field Name	Start index

13.3.2.2.9.1 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.

13.3.2.2.10 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 13-19.

Figure 13-19. Format of the Get Endpoint List Request Command

Octets	1
Data Type	Unsigned 8-bit integer
Field Name	Start index

13.3.2.2.10.1 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.

13.3.2.3 Commands Generated

When a device implements the *touchlink commissioning* cluster at the ZCL server side, it shall be able to generate the commands listed in Table 13-15.

Table 13-15. Commands Generated by the Server Side of the Touchlink Commissioning Cluster

	Command Identifier Field Value	Description	Mandatory/ Optional	Reference
Touchlink	0x01	Scan response	Mandatory	13.3.2.3.1
	0x03	Device information response	Mandatory	13.3.2.3.2
	0x11	Network start response	Mandatory	13.3.2.3.3
	0x13	Network join router response	Mandatory	13.3.2.3.4
	0x15	Network join end device response	Mandatory	13.3.2.3.5
	All other values in the range 0x00 – 0x3f	Reserved	-	-
Utility	0x40	Endpoint information	Mandatory	13.3.2.3.6
	0x41	Get group identifiers response	Mandatory	13.3.2.3.7
	0x42	Get endpoint list response	Mandatory	13.3.2.3.8
	All other values in the range 0x40 – 0xff	Reserved	-	-

13.3.2.3.1 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.

Note: If the Profile Interop bit of the Touchlink Information field of the received Scan Request command is set to zero, the device may choose to represent its device information in the form of ZLL device information to support legacy devices. If this bit is set to one, the device shall use the device information as given in its simple descriptors.[158]

This inter-PAN command shall be formatted according to the general inter-PAN frame format 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 13-20.

Figure 13-20. Format of the Scan Response Command Frame

Octets	Data Type	Field Name
4	Unsigned 32-bit integer	Inter-PAN transaction identifier
1	Unsigned 8-bit integer	RSSI correction
1	8-bit bitmap	ZigBee information

[158] CCB 2115

Octets	Data Type	Field Name
1	8-bit bitmap	Touchlink information
2	16-bit bitmap	Key bitmask
4	Unsigned 32-bit integer	Response identifier
8	IEEE address	Extended PAN identifier
1	Unsigned 8-bit integer	Network update identifier
1	Unsigned 8-bit integer	Logical channel
2	Unsigned 16-bit integer	PAN identifier
2	Unsigned 16-bit integer	Network address
1	Unsigned 8-bit integer	Number of sub-devices
1	Unsigned 8-bit integer	Total group identifiers
0/1	Unsigned 8-bit integer	Endpoint identifier
0/2	Unsigned 16-bit integer	Profile identifier
0/2	Unsigned 16-bit integer	Device identifier
0/1	Unsigned 8-bit integer	Version
0/1	Unsigned 8-bit integer	Group identifier count

13.3.2.3.1.1 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.

13.3.2.3.1.2 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.

13.3.2.3.1.3 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 13-21.

Figure 13-21. Format of the ZigBee Information Field

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

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.

13.3.2.3.1.4 Touchlink Information Field

The *Touchlink information* field is 8-bits in length and shall be formatted as illustrated in Figure 13-22.

Figure 13-22. Format of the Scan Response Touchlink Information Field

Bits: 0	1	2-3	4	5	6	7
Factory new	Address assignment	Reserved	Touchlink initiator	Touchlink priority request	Reserved	Profile Interop[159]

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.

The Profile Interop subfield is 1 bit in length and specifies which profile the device implements. If the ZLL profile is implemented, this bit shall be set to 0. In all other case (Profile Interop / ZigBee 3.0), this bit shall be set to 1.[160]

13.3.2.3.1.5 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 $0 \leq i \leq 15$.

13.3.2.3.1.6 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.

13.3.2.3.1.7 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 *touchlink 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 *touchlink 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 *touchlink 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.

13.3.2.3.1.8 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 touchlink information indicates the device to be in factory new mode, this field shall contain the value 0x00.

13.3.2.3.1.9 Logical Channel Field

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

[159] CCB 2115
[160] CCB 2115

If the *factory new* subfield of the *touchlink 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 *touchlink 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 *touchlink 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.

13.3.2.3.1.10 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 *touchlink 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 *touchlink 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 *touchlink 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.

13.3.2.3.1.11 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 touchlink information indicates the device to be in factory new mode, this value shall be set to 0xffff.

13.3.2.3.1.12 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.

13.3.2.3.1.13 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.

13.3.2.3.1.14 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.

13.3.2.3.1.15 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.

13.3.2.3.1.16 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.

13.3.2.3.1.17 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; 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.

13.3.2.3.1.18 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.

13.3.2.3.2 Device Information Response Command Frame

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

Note: If the Profile Interop bit of the Touchlink Information field of the Scan Request command received at the beginning of the current touchlink exchange is set to zero, the device may choose to represent its device information in the form of ZLL device information to support legacy devices. If this bit is set to one, the device shall use the device information as given in its simple descriptors.[161]

This inter-PAN command shall be formatted according to the general inter-PAN frame format 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 13-23.

Figure 13-23. Format of the Device Information Response Command Frame

Octets	4	1	1	1	(n*16)
Data Type	uint32	uint8	uint8	uint8	Variable
Field Name	Inter-PAN transaction identifier	Number of sub devices	Start index	Device information record count	Device information record (see Figure 13-24)

Figure 13-24. Format of the Device Information Record Field

Octets	8	1	2	2	1	1	1
Data Type	IEEE address	uint8	uint16	uint16	uint8	uint8	uint8
Field Name	IEEE address	Endpoint identifier	Profile identifier	Device identifier	Version	Group identifier count	Sort

13.3.2.3.2.1 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.

13.3.2.3.2.2 Number of Sub-devices Field

[161] CCB 2115

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.

13.3.2.3.2.3 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.

13.3.2.3.2.4 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.

13.3.2.3.2.5 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.

13.3.2.3.2.6 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.

13.3.2.3.2.7 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.

13.3.2.3.2.8 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.

13.3.2.3.2.9 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; the most significant 4 bits shall be set to 0x0.

13.3.2.3.2.10 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.

13.3.2.3.2.11 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.

13.3.2.3.3 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 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 13-25.

Figure 13-25. Format of the Network Start Response Command Frame

Octets	4	1	8	1	1	2
Data Type	uint32	uint8	IEEE address	uint8	uint8	uint16
Field Name	Inter-PAN transaction identifier	Status	Extended PAN identifier	Network update identifier	Logical channel	PAN identifier

13.3.2.3.3.1 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.

13.3.2.3.3.2 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 13-16.

Table 13-16. Values of the Status Field of the Network Start Response Command Frame

Status Field Value	Description
0x00	Success
0x01	Failure
0x02 – 0xff	Reserved

13.3.2.3.3.3 Extended PAN Identifier Field

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

13.3.2.3.3.4 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.

13.3.2.3.3.5 Logical Channel Field

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

13.3.2.3.3.6 PAN Identifier Field

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

13.3.2.3.4 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 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 13-26.

Figure 13-26. Format of the Network Join Router Response Command Frame

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

13.3.2.3.4.1 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.

13.3.2.3.4.2 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 13-17.

Table 13-17. Values of the Status Field of the Network Join Router Response Command Frame

Status Field Value	Description
0x00	Success
0x01	Failure
0x02 – 0xff	Reserved

13.3.2.3.5 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 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 13-27.

Figure 13-27. Format of the Network Join End Device Response Command Frame

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

13.3.2.3.5.1 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.

13.3.2.3.5.2 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 13-18.

Table 13-18. 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

13.3.2.3.6 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 command.

This command shall be formatted as illustrated in Figure 13-28.

Figure 13-28. Format of the Endpoint Information Command

Octets	8	2	1	2	2	1
Data Type	IEEE address	uint16	uint8	uint16	uint16	uint8
Field Name	IEEE address	Network address	Endpoint identifier	Profile identifier	Device identifier	Version

13.3.2.3.6.1 IEEE Address Field

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

13.3.2.3.6.2 Network Address Field

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

13.3.2.3.6.3 Endpoint Identifier Field

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

13.3.2.3.6.4 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.

13.3.2.3.6.5 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.

13.3.2.3.6.6 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; the most significant 4 bits shall be set to 0x0.

13.3.2.3.7 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 13-29.

Figure 13-29. Format of the Get Group Identifiers Response Command

Octets	1	1	1	(*n**3)
Data Type	uint8	uint8	uint8	Variable
Field Name	Total	Start index	Count	Group information record list

13.3.2.3.7.1 Total Field

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

13.3.2.3.7.2 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.

13.3.2.3.7.3 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.

13.3.2.3.7.4 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 13-30.

Figure 13-30. Format of a Group Information Record Entry

Octets	2	1
Data Type	uint16	uint8
Field Name	Group identifier	Group type

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.

13.3.2.3.8 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 13-31.

Figure 13-31. Format of the Get Endpoint List Response Command

Octets	1	1	1	(*n**8)
Data Type	uint8	uint8	uint8	Variable
Field Name	Total	Start index	Count	Endpoint information record list (see Figure 13-32)

Figure 13-32. Format of an Endpoint Information Record Entry

Octets	2	1	2	2	1
Data Type	uint16	uint8	uint16	uint16	uint8
Field Name	Network address	Endpoint identifier	Profile identifier	Device identifier	Version

13.3.2.3.8.1 Total Field

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

13.3.2.3.8.2 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.

13.3.2.3.8.3 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.

13.3.2.3.8.4 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.

13.3.2.3.8.5 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.

13.3.2.3.8.6 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.

13.3.2.3.8.7 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.

13.3.2.3.8.8 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; the most significant 4 bits shall be set to 0x0.

13.3.3 Client

13.3.3.1 Attributes

The client has no attributes.

13.3.3.2 Commands Received

The client receives the cluster specific response commands listed in Table 13-19. These commands are detailed in 13.3.2.2.

Table 13-19. 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; otherwise Optional
	0x42	Get endpoint list response	Mandatory if *get endpoint list request* command is generated; otherwise Optional

13.3.3.3 Commands Generated

The client generates the cluster specific commands listed in Table 13-20. These commands are detailed in 13.3.2.3.

Table 13-20. 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

	Identifier	Description	Usage
	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

13.3.4 Functional Description

13.3.4.1 Profile Identifier

Those commands in the touchlink commissioning command set shall be sent using the profile identifier, 0xc05e whereas those commands in the commissioning utility command set shall sent using the ZHA profile identifier, 0x0104.

13.3.4.2 Constants

The constants that define the characteristics of touchlink commissioning are listed in Table 13-21.

Table 13-21. Touchlink Commissioning Constants

Constant	Description	Value
aplcInterPANTransIdLifetime	The maximum length of time an inter-PAN transaction identifier remains valid.	8s
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

13.3.4.3 Attributes

Touchlink commissioning defines internal attributes required to allow a device to manage the way it operates. These attributes are summarized in Table 13-22.

Table 13-22. Touchlink Commissioning Attributes

Attribute	Type	Ref	Default
aplFreeNwkAddrRangeBegin	Unsigned 16-bit integer	10.1.1.3.1	0x0001

aplFreeNwkAddrRangeEnd	Unsigned 16-bit integer	10.1.1.3.2	0xfff7
aplFreeGroupIDRangeBegin	Unsigned 16-bit integer	10.1.1.3.3	0x0001
aplFreeGroupIDRangeEnd	Unsigned 16-bit integer	10.1.1.3.4	0xfeff

13.3.4.3.1 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 network through classical ZigBee joining mechanisms, this attribute should be set to 0x0000.

13.3.4.3.2 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 network through classical ZigBee joining mechanisms, this attribute should be set to 0x0000.

13.3.4.3.3 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 network through classical ZigBee joining mechanisms, this attribute should be set to 0x0000.

13.3.4.3.4 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 network through classical ZigBee joining mechanisms, this attribute should be set to 0x0000.

13.3.4.4 Device Information Table

Each device supporting touchlink commissioning shall contain a *device information table* that holds the necessary (static) application information that is exchanged during touchlink device discovery. 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 Figure 13-33.

Figure 13-33. 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

13.3.4.4.1 IEEE Address Field

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

13.3.4.4.2 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.

13.3.4.4.3 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.

13.3.4.4.4 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.

13.3.4.4.5 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.

13.3.4.4.6 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.

13.3.4.4.7 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.

13.3.4.5 Inter-PAN frame format[162]

When using the inter-PAN frame format for touchlink 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 below.

[162] CCB 2105

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[163] 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 13.3
ZCL payload	Command payload	Variable	See clause 13.3
MAC footer	Frame check sequence	2	As appropriate for the frame

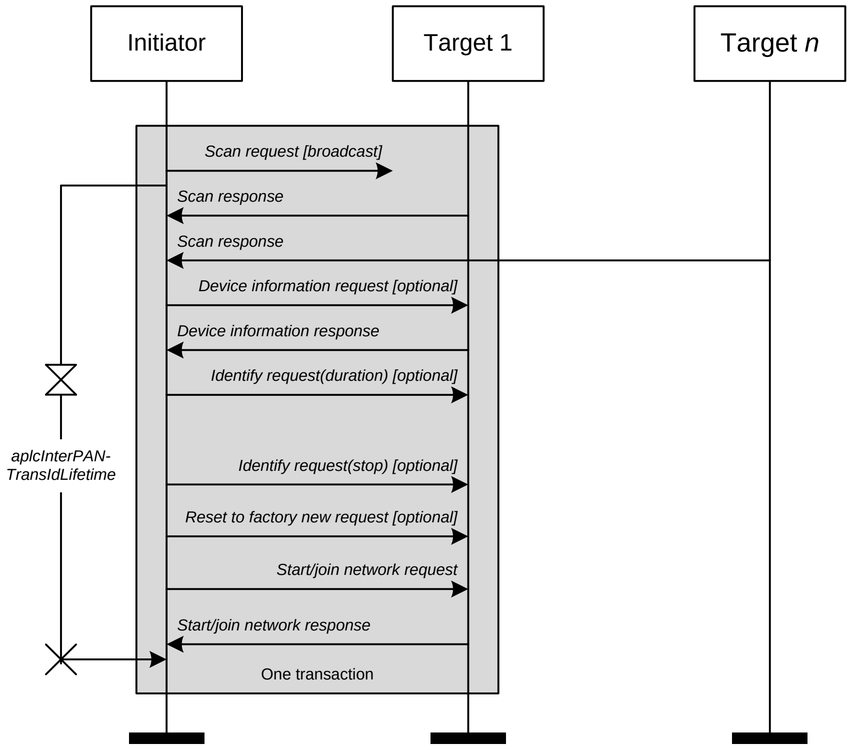
Figure 13-34. General format of an inter-PAN frame

13.3.4.6 Inter-PAN Transaction Identifier

All *touchlink commissioning* cluster 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 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 13-35) 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.

Figure 13-35. Scope of a touchlink commissioning inter-PAN transaction


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

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.

[163] When using the inter-PAN command frames defined in the Touchlink Commissioning cluster, the manufacturer specific sub-field of the ZCL header frame control field shall be set to 0.

13.3.4.7 Commissioning Scenarios

Touchlink commissioning between 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.

For detailed information on touchlink commissioning, see the Base Device Behavior Specification.

13.3.4.8 Address Assignment

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

13.3.4.8.1 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 touchlink network address space is defined in the range (0x0001 ⋯ 0xfff7). 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.

13.3.4.8.2 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. 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. 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.

13.3.4.9 Network Update

13.3.4.9.1 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 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.

13.3.4.9.2 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 TBD. 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.

13.3.4.10 Frequency Agility

Touchlink 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 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 );
```

13.3.4.11 Security

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

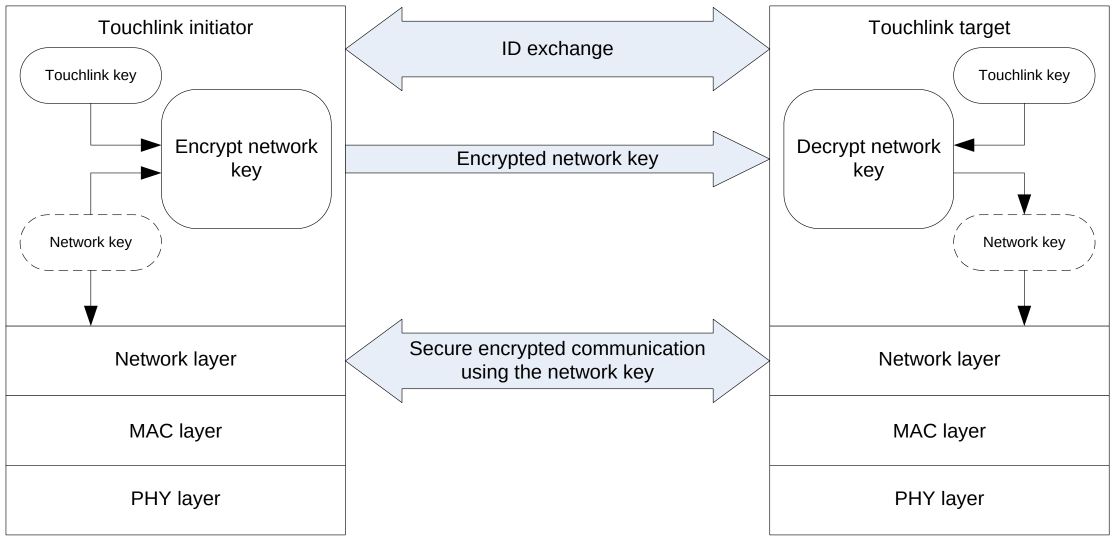
In this clause concatenation of strings is represented by the "||" symbol.

13.3.4.11.1 Transferring the Network Key during Touchlink Commissioning

The touchlink security architecture is based on using a fixed secret key, known as the touchlink key, which shall be stored in each device. During touchlink commissioning, all devices use the touchlink 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 13-36.

Figure 13-36. Overview of Touchlink 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 13-23, 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 13-23. Key Encryption Algorithms

Key index	Key description	Algorithm
0	Development key	See 13.3.4.11.4
1-3	Reserved	-
4	Master key	See 13.3.4.11.5
5-14	Reserved for future use	-
15	Certification key	See 13.3.4.11.5

13.3.4.11.2 Transferring the Network Key during Classical ZigBee Commissioning

During classical ZigBee commissioning where a device is being joined to a network without a trust center, a pre-installed link key is used to secure the transfer of the network key when authenticating. The pre-installed link key is a secret shared by all certified devices. It will be distributed only to certified manufacturers and is bound with a safekeeping contract.

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, devices shall try to decode the APS message using the known default trust center link key.

13.3.4.11.3 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).

13.3.4.11.4 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.

13.3.4.11.5 Key Index 4 and 15

13.3.4.11.5.1 Key Usage

The touchlink security details described in this section apply to key index 4 and 15 of Table 13-23. The secure NWK key transport methods indicted by key index 4 and 15 use the same algorithm, as described in section 13.3.4.11.5.2. However, they differ in the type of key they use for NWK key protection.

13.3.4.11.5.1.1 Master Key (key index 4)

The touchlink master key is a secret shared by all certified devices. It will be distributed only to certified manufacturers and is bound with a safekeeping contract.

The device using the touchlink 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 13.3.2.3.1).

13.3.4.11.5.1.2 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 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 13.3.2.3.1).

The certification key may also be used during the development phase of products. However, commercial products shall not use nor indicate having support for the certification key.

13.3.4.11.5.2 Algorithm

13.3.4.11.5.2.1 Encrypting Network Keys for Touchlink Initiator

The touchlink initiator shall perform the following steps to encrypt the network key and transport it to the touchlink target:

- Exchange of transaction identifier and response identifier as part of the touchlink procedure.
- Derive the ephemeral transport key (see Figure 13-37) from the transaction identifier, response identifier and touchlink master or certification key, as described in 13.3.4.11.5.2.3.
- Encrypt the network key using the calculated transport key and the AES ECB mode, as described in 13.3.4.11.5.2.3.
- Transmit the encrypted network key to the touchlink target as part of the touchlink procedure.

13.3.4.11.5.2.2 Decrypting network keys for touchlink target

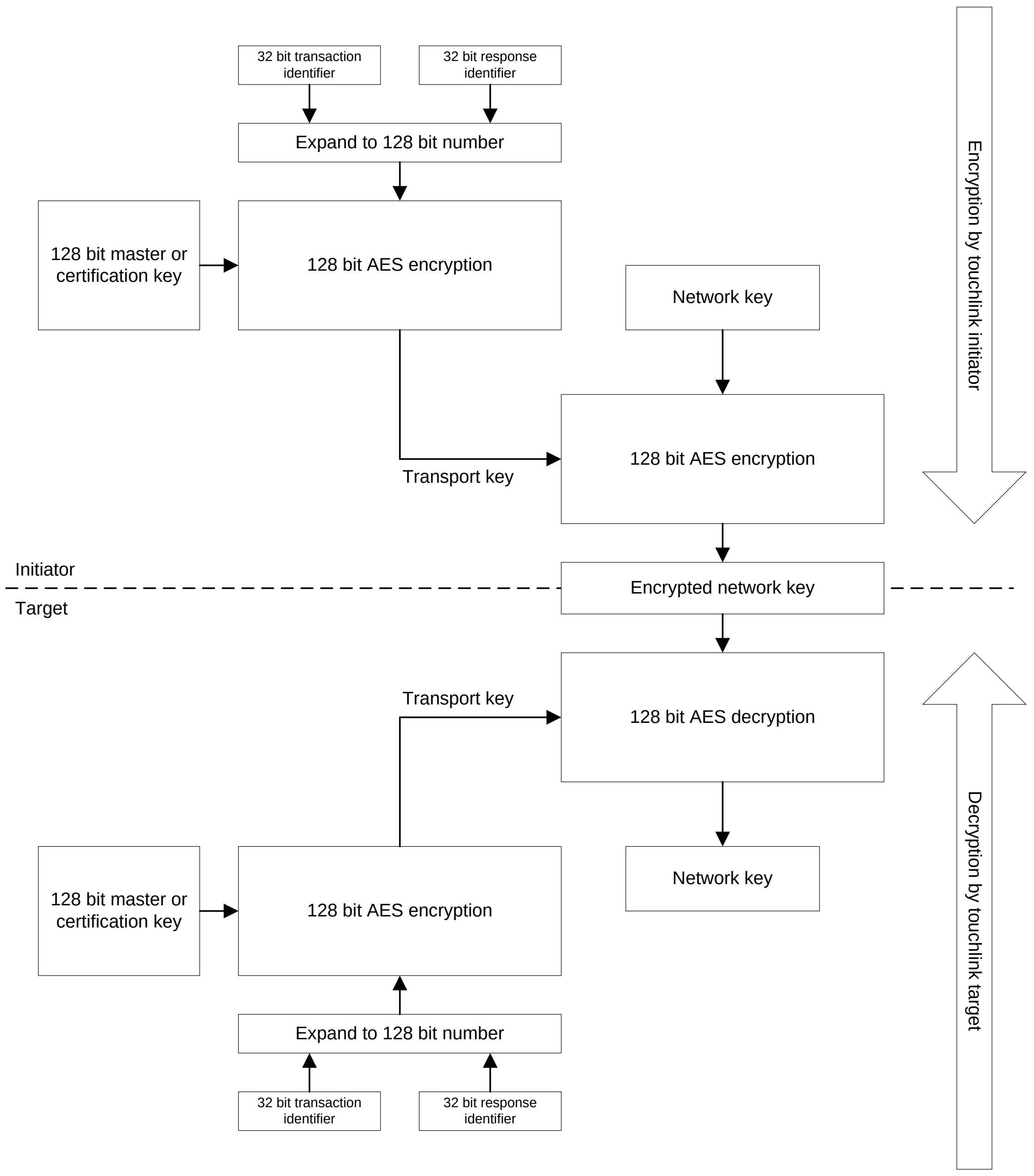
The touchlink target shall perform the following steps to decrypt the network key received from the touchlink initiator:

- Exchange of transaction identifier and response identifier as part of the touchlink procedure.
- Receive the encrypted network key as part of the touchlink procedure.
- Derive the transport key (see Figure 13-37) from the transaction identifier, response identifier and touchlink master or certification key, as described in 13.3.4.11.5.2.3.
- Decrypt the received encrypted network key by using the calculated transport key and the AES ECB mode, as described in 13.3.4.11.5.2.3.
- Store the received network key in the NIB parameter of the touchlink target.

13.3.4.11.5.2.3 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 13-37.

Figure 13-37. Steps Required to Encrypt/Decrypt the Network Key


32 bit transaction identifier
32 bit response identifier
Expand to 128 bit number
128 bit master or certification key
128 bit AES encryption
Network key
Transport key
128 bit AES encryption
Encryption by touchlink initiator
Initiator
Encrypted network key
Target
Transport key
128 bit AES decryption
Network key
128 bit master or certification key
128 bit AES encryption
Expand to 128 bit number
32 bit transaction identifier
32 bit response identifier
Decryption by touchlink target

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 touchlink master or certification key

The encryption of the network key is performed by the following processing steps:

8. 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.

9. Calculate the transport key by executing the 128 bit AES encryption with the expanded 128 bit number obtained from step 1 as *plaintext*, and touchlink master or certification key as *key*.
10. Encrypt the network key by executing the 128 bit AES encryption using the network key as *plaintext* and the transport key obtained from step 2 as *key*.

The decryption of the network key is performed by the following processing steps:

11. Merge and expand the transaction identifier and response identifier into a 128 bit number, as described in step 1.
12. Calculate the transport key by executing the 128 bit AES encryption with the expanded 128 bit number obtained from step 4 used as *plaintext*, and touchlink master or certification key as *key*.
13. Decrypt the network key by executing the 128 bit AES decryption with the transport key obtained from step 5 as *key* and the encrypted network key as *ciphertext*.

All AES functions used in steps 2, 3, and 5 above shall use AES encryption in ECB mode and the AES function in step 6 shall use AES decryption in ECB mode.

13.3.4.11.6 Touchlink Security Test Vectors

This annex provides sample test vectors for the touchlink security specification (as defined in sub-clause 13.3.4.11), in order to assist in building interoperable security implementations.

13.3.4.11.6.1 Touchlink initiator operation

Touchlink 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

13.3.4.11.6.2 Touchlink target operation

Touchlink 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.

CHAPTER 14 RETAIL

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

14.1 General Description

14.1.1 Introduction

The clusters specified in this chapter are for use typically in retail applications, but may be used in any application domain.

14.1.2 Cluster List

This section lists the clusters specified in this chapter and gives examples of typical usage for the purpose of clarification.

The clusters specified in this chapter are listed in Table 14-1.

Table 14-1. Clusters Specified in this Chapter

ID	Cluster Name	Description
0x0617	Retail Tunnel Cluster	Interface for manufacturer specific information to be exchanged
0x0022	Mobile Device Configuration Cluster	Interface to manage mobile devices in a network
0x0023	Neighbor Cleaning Cluster	Interface to manage mobile devices in a network
0x0024	Nearest Gateway Cluster	Interface to enable communication of nearest gateway to devices

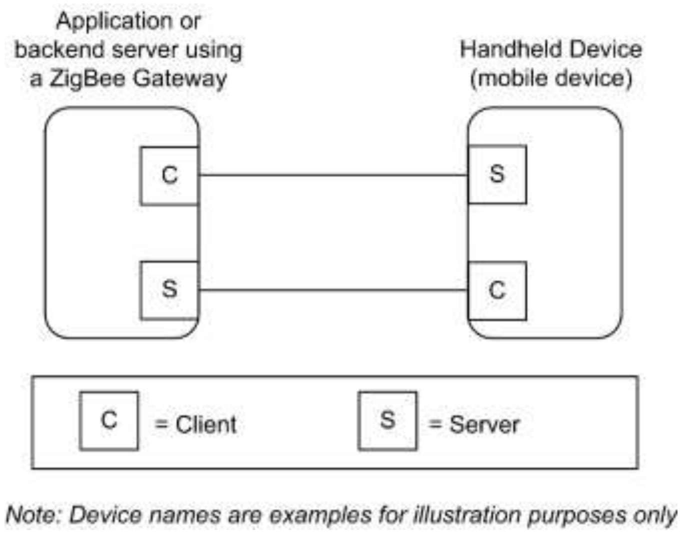
14.2 Retail Tunnel (MSP Tunnel)

14.2.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface for transferring information encoded through a specific Manufacturer specific Profile from a device (e.g., a backend application using a gateway) to a handheld device (e.g., the Retail HHD). The messages that are transferred use a transfer APDU command as for other tunneling clusters defined (e.g., 11073 Protocol tunnel, or ISO 7818 tunnel).

Figure 14-1. Typical Usage of the Retail Tunnel Cluster


Application or backend server using a ZigBee Gateway
Handheld Device (mobile device)
C
S
S
C
C = Client
S = Server
Note: Device names are examples for illustration purposes only

14.2.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

14.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	RTUN	Type 1 (client to server)

14.2.1.3 Cluster Identifiers

Identifier	Name
0x0617	Retail Tunnel

14.2.2 Server

14.2.2.1 Dependencies

This cluster may leverage on the Partition cluster in order to carry payloads not fitting into a single ZCL payload.

14.2.2.2 Attributes

The currently defined attributes for this cluster are listed in Table 14-2.

Table 14-2. Attributes of the Retail Tunnel cluster

Id	Name	Type	Range	Acc	Def	M/O
0x0000	*ManufacturerCode*	uint16	0x1000 – 0x10ff	R	-	M
0x0001	*MSProfile*	uint16	0xC000 – 0xFFFF	R	-	M

14.2.2.2.1 *ManufacturerCode* Attribute

The *ManufacturerCode* attribute specifies the manufacturer code relating the manufacturer of the device. This attribute can be used to match the proper protocol associated to the manufacturer of the device and tunneled through this cluster. See [Z12] Manufacturer Code Database.

14.2.2.2.2 *MSProfile* Attribute

The *MSProfile* attribute specifies the manufacturer specific profile used in the tunneled messages carried by the Transfer APDU commands. The *MSProfile* attribute can be used to have the information of the proper protocol used by the communication entities supporting the MSP Tunnel cluster in order to properly decode the messages tunneled in this cluster.

14.2.2.3 Commands Received

Table 14-3 lists the cluster-specific commands that are received by the server.

Table 14-3. Cluster-specific Commands Received by the Server

Command identifier field value	Description	Mandatory / Optional
0x00	Transfer APDU	M

14.2.2.3.1 Transfer APDU Command

14.2.2.3.1.1 Payload Format

The Transfer APDU command shall be formatted as illustrated in Figure 14-2.

Figure 14-2. Format of the Transfer APDU Command

Bits	**Variable**
Data Type	Octet String
Field Name	APDU

14.2.2.3.1.2 APDU Field

The APDU field is of variable length and is an APDU as defined in the *MSProfile* attribute of the Manufacturer indicated by the *ManufacturerCode* attribute.

14.2.2.3.1.3 When Generated

This command is generated when a message has to be transferred across a MSP tunnel. The message can be only decoded by the recipient entity if it is provided by the proper decodes of the Manufacturer specific profile as defined in [Z7].

14.2.2.3.1.4 Effect on Receipt

On receipt of this command, a device shall process the APDU according to the specific MSP transported.

14.2.2.4 Commands Generated

No cluster-specific commands are generated by the server cluster.

14.2.3 Client

The client has no dependencies, no cluster specific attributes.The client does not receive any cluster-specific commands. The client generates the cluster-specific commands detailed in 14.2.2.3.

14.3 Mobile Device Configuration

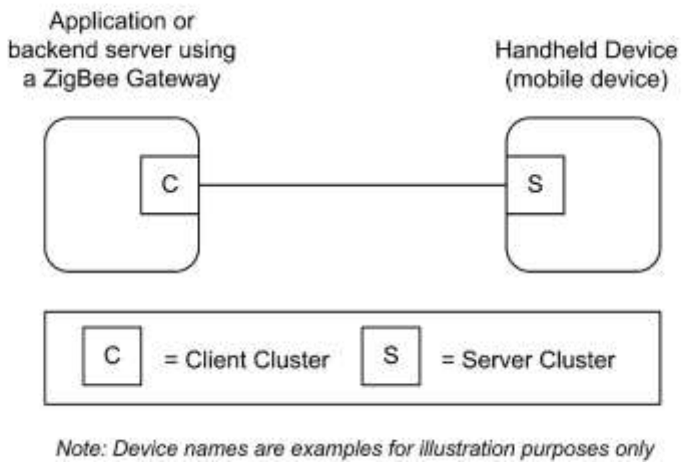
14.3.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface to enable the management of mobile devices in a network.

If a stack supports neighbor entry aging, the mobile device will be able to use this cluster to refresh the information in the parent/neighbor. An application will be also able to configure aging timeout (using the Neighbor cleaning cluster) greater than *KeepAliveTime*, managing in this way the timeout used for cleaning neighbor table setting appropriate value. Besides, *Rejoin timeout* can be used to allow the device force a rejoin and then allow the mobile device solution to work with stacks not supporting the cleaning of the neighbor tables.

Figure 14-3. Typical Usage of the Mobile Device Configuration Cluster


Application or backend server using a ZigBee Gateway
Handheld Device (mobile device)
C
S
C = Client Cluster
S = Server Cluster
Note: Device names are examples for illustration purposes only

14.3.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

14.3.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	MOBCFG

14.3.1.3 Cluster Identifiers

Identifier	Name
0x0022	Mobile Device Configuration

14.3.2 Server

14.3.2.1 Dependencies

This cluster should be supported by devices that are mobile in the network. The devices building the network infrastructure should use the Neighbor Cleaning Cluster to manage the loss of the mobile devices from the radio range.

14.3.2.2 Attributes

The currently defined attributes for this cluster are listed in Table 14-4.

Table 14-4. Attributes of the Mobile Device Cleaning Cluster

Identifier	Name	Type	Range	Acc	Unit	Default	M/O
0x0000	*KeepAliveTime*	uint16	0x0001- 0xFFFF	RW	Seconds	15 seconds (0x000F)	M
0x0001	*RejoinTimeout*	uint16	0x0000- 0xFFFF	RW	Seconds	0xFFFF (Never)	M

14.3.2.2.1 KeepAliveTime Attribute

The *KeepAliveTime* attribute specifies the time period to elapse before a mobile device send a Keep Alive Notification message to the manager of the network (e.g. application backend servers using a gateway). Please note that a value of this attribute equal to 0xFFFF means that the mobile device shall not send *KeepAliveNotification* messages. This attribute is used to “refresh” neighbor table information on its parent devices, avoiding expiration or aging of the correspondent entry.

14.3.2.2.2 RejoinTimeout Attribute

The *RejoinTimeout* attribute specifies the time after which the device shall perform a secure network rejoin to clean the entries in the neighbor table for parent devices not cleaning them with the Neighbor Cleaning Cluster. Please note that a value of this attribute equal to 0xFFFF means that the mobile device is not requested to perform the network Rejoin to clean the mesh. (Note: The mobile device may choose to transmit a Network Leave frame to the short address being cleaned.)

14.3.2.3 Commands Received

No cluster-specific commands are received by the server side of this cluster.

14.3.2.4 Commands Generated

Table 14-5 lists cluster-specific commands that are generated by the server.

Table 14-5. Cluster-specific Commands Generated by the Server

Command Id	Description	M/O
0x00	*Keep Alive Notification*	M

14.3.2.4.1 Keep Alive Notification Command

14.3.2.4.1.1 Payload Format

The Keep Alive Notification command shall be formatted as illustrated in Figure 14-4.

Figure 14-4. Format of the Keep Alive Notification Command

Bits	**Variable**	**Variable**
Data Type	uint16	uint16
Field Name	*KeepAliveTime*	*RejoinTimeout*

14.3.2.4.1.1.1 KeepAliveTime Field

This field corresponds to the *KeepAliveTime* attribute.

14.3.2.4.1.1.2 RejoinTimeout Field

This field corresponds to the *RejoinTimeout* attribute.

14.3.2.4.1.2 When Generated

This command is generated when a time greater than *KeepAliveTime* attribute elapses.

14.3.2.4.1.3 Effect on Receipt

On receipt of this command, a parent or neighbor device shall refresh neighbor table information on the mobile node sending the Keep Alive Notification by resetting the timers managing the expiration of the entries in the neighbors table.

14.3.3 Client

The client has no dependencies, no cluster specific attributes. The client receives the commands specified in section 14.2.2.4. The client does not generate any cluster-specific commands.

14.4 Neighbor Cleaning

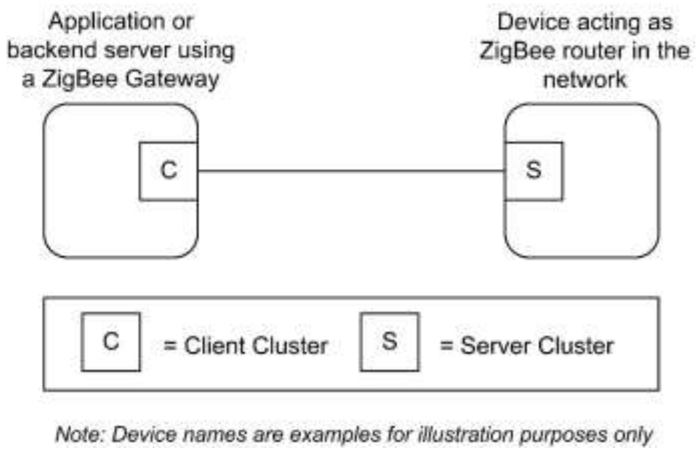
14.4.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface to enable the management of mobile devices in a network.

If a stack supports neighbor entry aging, the mobile device will be able to use this cluster to clean the information in the parent/neighbor. An application will be able to configure the aging timeout greater than a *KeepAliveTime* (attribute supported by a mobile device), managing in this way the timeout used for cleaning neighbor table setting appropriate value.

Figure 14-5. Typical Usage of the Neighbor Cleaning Cluster


Application or backend server using a ZigBee Gateway
Device acting as ZigBee router in the network
C
S
C = Client Cluster
S = Server Cluster
Note: Device names are examples for illustration purposes only

14.4.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

14.4.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	NBCLEAN

14.4.1.3 Cluster Identifiers

Identifier	Name
0x0023	Neighbor Cleaning

14.4.2 Server

14.4.2.1 Dependencies

This cluster should be supported by devices that are acting as routers for Mobile devices in the network; besides, the mobile devices within the network infrastructure (e.g., Hand Held devices or Mobile phones) should use the Mobile Device Configuration Cluster to take advantage of the mobility feature.

14.4.2.2 Attributes

The currently defined attributes for this cluster are listed in the following table.

Table 14-6. Attributes of the Neighbor Cleaning Cluster

Id	Name	Type	Range	Acc	Unit	Default	M/O
0x0000	*NeighborCleaningTimeout*	uint16	0x0001 - 0xFFFF	RW	Seconds	30 seconds (0x001E)	M

14.4.2.2.1 NeighborCleaningTimeout Attribute

The *NeighborCleaningTimeout* attribute specifies the time period to elapse without receiving any messages from a neighbor device (router or end device) which is a mobile device, before cleaning its neighbor table entry. (Note: The cleaning device may choose to transmit a Network Leave frame to the short address being cleaned.)

14.4.2.3 Commands Received

Table 14-7 lists cluster-specific commands which are received by the server side of this cluster.

Table 14-7. Cluster-specific Commands Generated by the Server

Command Id	Description	M/O
0x00	*PurgeEntries*	M

14.4.2.3.1 PurgeEntries Command

14.4.2.3.1.1 Payload Format

The *PurgeEntries* command has no payload.

14.4.2.3.1.2 When Generated

This command is generated by the manager of the network supporting the mobile devices in order to force the cleaning of the neighbor table entries.

14.4.2.3.1.3 Effect on Receipt

On receipt of this command, a parent or neighbor device should clean the neighbor tables to delete aged entries; please notice that this feature can be executed only if enabled by the stack.

14.4.2.4 Commands Generated

No cluster-specific commands are generated by the server.

14.4.3 Client

The client has no dependencies and no cluster specific attributes. The client does not receive any cluster-specific commands. The client does generate the cluster-specific commands specified in 14.4.2.3.

14.5 Nearest Gateway

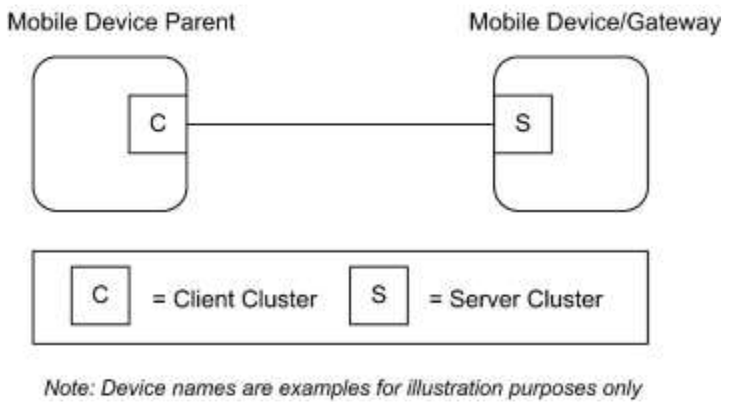
14.5.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

This cluster provides an interface to enable the dissemination of "nearest gateway" information.

Based on MTORR information initiated by gateway devices (concentrator), the remaining routers in the network can determine which gateway is closest based on path cost, i.e., the "nearest gateway." The cluster allows that information to be communicated to devices in the network that need that information.

Figure 14-6. Typical Usage of the Nearest Gateway Cluster



14.5.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

14.5.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	NEARGW

14.5.1.3 Cluster Identifiers

Identifier	Name
0x0024	Nearest Gateway

14.5.2 Server

14.5.2.1 Dependencies

This cluster should be supported by devices that are mobile in the network and, optionally, gateway devices.

14.5.2.2 Attributes

The currently defined attributes for this cluster are listed in the following table.

Table 14-8. Attributes of the Nearest Gateway Cluster

Id	Name	Type	Range	Acc	Default	M/O
0x0000	*Nearest Gateway*	16-bit NWK address	0x0000- 0xFFF8	RW	0x0000	M
0x0001	*New Mobile Node*	16-bit NWK address	0x0000- 0xFFF8	W	0x0000	M

14.5.2.2.1 Nearest Gateway Attribute

The *Nearest Gateway* attribute specifies the gateway that is nearest in terms of path cost.

14.5.2.2.2 New Mobile Node Attribute

The *New Mobile Node* attribute specifies the new mobile node that joined the server.

14.5.2.3 Commands Received

No cluster-specific commands are received by the server side of this cluster.

14.5.2.4 Commands Generated

No cluster-specific commands are generated by the server side of this cluster.

14.5.3 Client

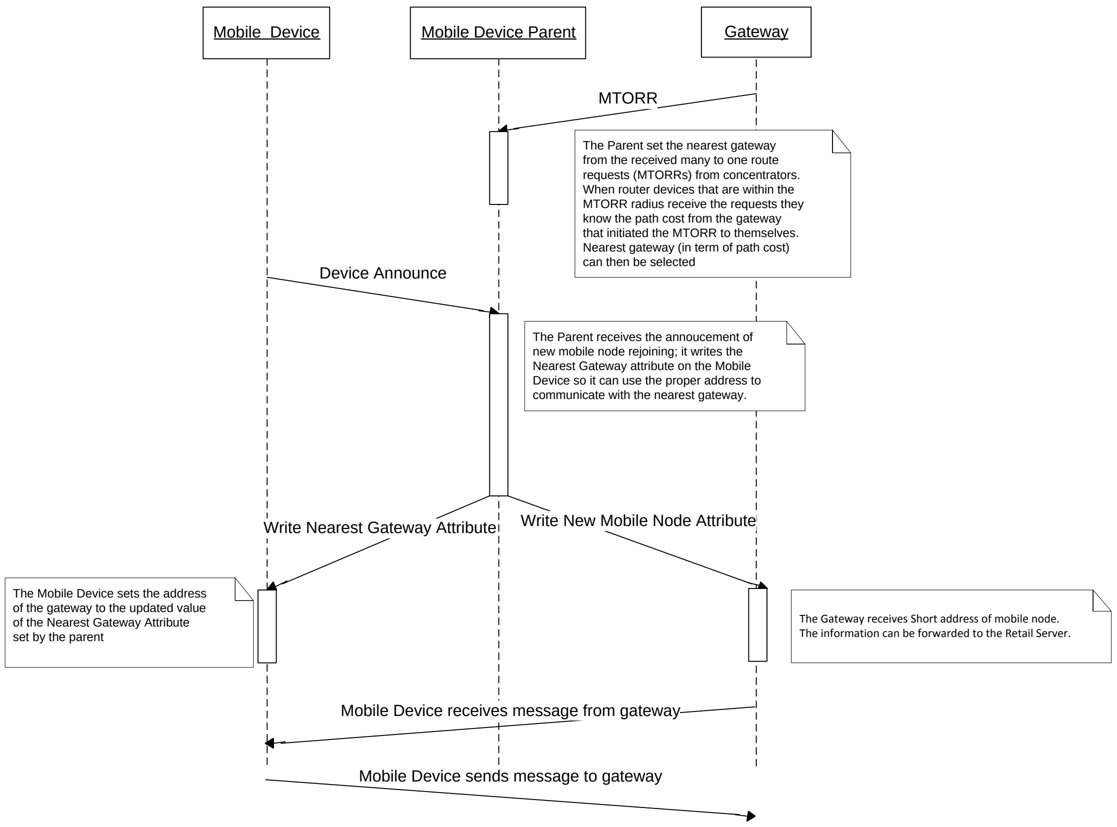
The client has no dependencies and no cluster specific attributes. The client does not receive nor generate any cluster-specific commands.

14.5.4 Examples of Use

Figure 14-7 describes an example of the possible use of the nearest gateway cluster.

20117

Figure 14-7. Sequence Diagram


Mobile Device
Mobile Device Parent
Gateway
MTORR
The Parent set the nearest gateway from the received many to one route requests (MTORRs) from concentrators. When router devices that are within the MTORR radius receive the requests they know the path cost from the gateway that initiated the MTORR to themselves. Nearest gateway (in term of path cost) can then be selected
Device Announce
The Parent receives the annoucement of new mobile node rejoining; it writes the Nearest Gateway attribute on the Mobile Device so it can use the proper address to communicate with the nearest gateway.
Write Nearest Gateway Attribute
Write New Mobile Node Attribute
The Mobile Device sets the address of the gateway to the updated value of the Nearest Gateway Attribute set by the parent
The Gateway receives Short address of mobile node. The information can be forwarded to the Retail Server.
Mobile Device receives message from gateway
Mobile Device sends message to gateway

20118

20119

CHAPTER 15 APPLIANCE

The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.

15.1 General Description

15.1.1 Introduction

The clusters specified in this chapter are for use typically in appliance management, but MAY be used in any application domain.

15.1.2 Cluster List

This section lists the clusters specified in this chapter and gives examples of typical usage for the purpose of clarification.

The clusters specified in this chapter are listed in Table 15-1.

Table 15-1. Appliance Management Clusters

Id	Cluster Name	Description
0x001b	EN50523 Appliance Control	Commands and attributes for controlling household appliances
0x0b00	EN50523 Appliance Identification	Commands and attributes for appliance information and device settings
0x0b02	EN50523 Appliance Events and Alerts	Commands and attributes for appliance events and alerts
0x0b03	EN50523 Appliance Statistics	Commands and attributes for appliance statistics

15.2 EN50523 Appliance Control

This section describes the EN50523 Appliance Control cluster.

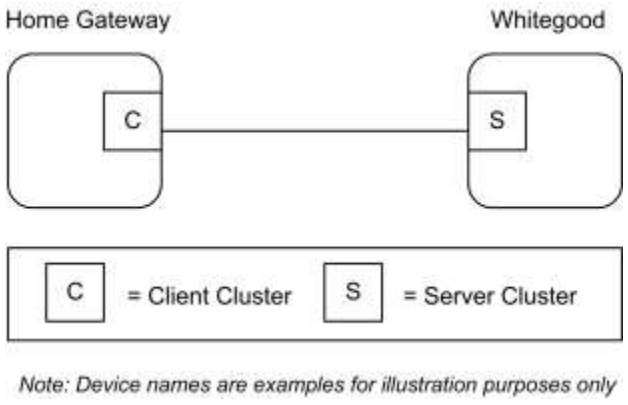
15.2.1 Overview

Please see section 2.2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc

This cluster provides an interface to remotely control and to program household appliances. Example of control is Start, Stop and Pause commands.

The status “read” and “set” is compliant to the EN50523 “Signal State” and “Execute Command” functional blocks. Appliances parameters (e.g., Duration and Remaining Time) have been added, since they were missing from the original specs.

Figure 15-1. Typical Usage of the Appliance Control Cluster


Home Gateway
Whitegood
C
S
C
= Client Cluster
S
= Server Cluster
Note: Device names are examples for illustration purposes only

Note: *Where a physical node supports multiple endpoints it will often be the case that many of these settings will apply to the whole node, that is, they are the same for every endpoint on the device. In such cases they can be implemented once for the node and mapped to each endpoint.*

15.2.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

15.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	APLNC	Type 2 (server to client)

15.2.1.3 Cluster Identifiers

Identifier	Name
0x001b	EN50523 Appliance Control

15.2.2 General Description

15.2.3 Server Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 256 attributes. Attribute identifiers are encoded such that the most significant byte specifies the attribute set and the least significant byte specifies the attribute within the set. The currently defined attribute sets are listed in Table 15-2.

Table 15-2. Appliance Control Attribute Set

Attribute Set Identifier	Description
0x00	Appliance Functions

15.2.3.1 Appliance Functions Attribute Set

The Appliance Functions attribute set contains the attributes summarized in Table 15-3.

These attributes control the Appliance cycle parameters. Each of them, as described below, corresponds to an Appliance internal status configuration.

Table 15-3. Attributes of the Appliance Functions Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0000	*StartTime*	uint16	0x0000 – 0xffff	RP	0x0000	M
0x0001	*FinishTime*	uint16	0x0000 – 0xffff	RP	0x0000	M
0x0002	*RemainingTime*	uint16	0x0000 – 0xffff	RP	0x0000	O

15.2.3.2 *StartTime* Attribute

StartTime attribute determines the time (either relative or absolute) of the start of the machine activity. Default format for Oven devices is absolute time. The default format for other appliances is relative time. *StartTime* SHOULD be set less than *FinishTime*.

Table 15-4 provides details about time encoding which is used for *StartTime* attribute organization.

Table 15-4. Time Encoding

Bit Range	Function	
0..5	Minutes ranging from 0 to 59	
6..7	Time encoding	
	Value	**Enumeration**
	0x0 0x1 0x2..0x3	RELATIVE ABSOLUTE Reserved
8..15	Hours ranging from 0 to 255 if RELATIVE encoding is selected 0 to 23 if ABSOLUTE encoding is selected	

15.2.3.3 *FinishTime* Attribute

FinishTime attribute determines the time (either relative or absolute) of the expected end of the machine activity. Default format for Oven is absolute time. The default format for other appliances is relative time. *FinishTime* SHOULD be set greater than *StartTime*.

FinishTime attribute exploits time encoding reported in Table 15-4.

15.2.3.4 *RemainingTime* Attribute

RemainingTime attribute determines the time, in relative format, of the remaining time of the machine cycle. It represents the time remaining to complete the machine cycle and it is updated only during the RUNNING state of the Appliance. During the other states of the Appliance *RemainingTime* attribute is indicated as the not valid value "0".

RemainingTime attribute exploits time encoding reported in Table 15-4.

15.2.4 Server Commands Received

The command IDs for the Appliance Control cluster are listed in Table 15-5.

Table 15-5. Cluster-specific Commands Received by the Server

Command Identifier Field Value	Description	M/O
0x00	Execution of a Command	O
0x01	Signal State	M
0x02	Write Functions	O
0x03	Overload Pause Resume	O
0x04	Overload Pause	O
0x05	Overload Warning	O

15.2.4.1 Execution of a Command

This basic message is used to remotely control and to program household appliances. Examples of control are START, STOP and PAUSE.

15.2.4.1.1 Payload Format

The Execution of a Command payload SHALL be formatted as illustrated in Figure 15-2.

Figure 15-2. Format of the Execution of a Command Payload

Octets	1
Data Type	enum8
Field Name	Command Identification

15.2.4.1.1.1 Payload Details

The *Command Identification* field: the command identification is an 8-bits in length field identifying the command to be executed. The enumeration used for this field SHALL match Table 15-6.

Table 15-6. Command Identification Values

Enumeration	Value	Description
START	0x01	Start appliance cycle

Enumeration	Value	Description
STOP	0x02	Stop appliance cycle
PAUSE	0x03	Pause appliance cycle
START SUPERFREEZING	0x04	Start superfreezing cycle
STOP SUPERFREEZING	0x05	Stop superfreezing cycle
START SUPERCOOLING	0x06	Start supercooling cycle
STOP SUPERCOOLING	0x07	Stop supercooling cycle
DISABLE GAS	0x08	Disable gas
ENABLE GAS	0x09	Enable gas
Manufacturer Specific	0x80..0xff	Manufacturer Specific

15.2.4.1.2 Effects on Receipt

On receipt of this command, the appliance SHALL execute the command given in the Command Identification field. The device application SHALL be informed of the imposed command (and potential personalized tasks could start, e.g., by means of a message to appliance Main Board controller).

After the command execution, the appliance SHALL generate a Signal State Notification with the new appliance state.

15.2.4.2 Signal State Command

This basic message is used to retrieve Household Appliances status. This command does not have a payload.

15.2.4.2.1 Effects on Receipt

On receipt of this command, the device SHALL generate a Signal State Response command.

15.2.4.3 Write Functions Command

This basic message is used to set appliance functions, i.e., information regarding the execution of an appliance cycle. Condition parameters such as start time or finish time information could be provided through this command. A function is mirrored by the cluster attribute that represents its current state. See Effect on Receipt below to understand the difference between writing a function and writing an attribute value.

15.2.4.3.1 Payload Format

The Write Functions command frame SHALL be formatted as illustrated in Format of the Write Functions Command Frame.

Figure 15-3. Format of the Write Functions Command Frame

Octets	Variable
Field Name	Write Functions record

Write Functions record SHALL be formatted as illustrated in Figure 15-4.

Figure 15-4. Format of the Write Functions Record Field

Octets	2	1	Variable
Data Type	uint16	enum8	Variable
Field Name	Function identifier (i.e., attribute identifier)	Function data type	Function data

15.2.4.3.2 Payload Details

The **Function identifier** field: the Function Identifier is 16-bits in length and SHALL contain the identifier of the function that is to be written.

The **Function data type** field: the function data type field SHALL contain the data type identifier of the attribute that is to be written.

The **Function data** field: the function data field is variable in length and SHALL contain the actual value of the function that is to be written.

15.2.4.3.3 Effects on Receipt

On receipt of this command, the appliance SHALL set the function given in the Function identifier field. The Function attribute is actually changed only when the appliance internal functions have been changed.

If attribute reporting is configured on some function attributes, an attribute reporting command is generated when the attribute, and therefore internal appliance function is actually modified. In case attribute reporting is not used, the correct execution of the Write Function command SHOULD be verified by using Read Attribute command to poll the written attribute.

15.2.4.4 Overload Pause Resume Command

This command SHALL be used to resume the normal behavior of a household appliance being in pause mode after receiving a Overload Pause command.

15.2.4.4.1 Payload Format

The Overload Pause Resume Command SHALL have no payload.

15.2.4.4.2 Effects on Receipt

On receipt of this command, the appliance SHALL resume its operations.

15.2.4.5 Overload Pause Command

This command SHALL be used to pause the household appliance as a consequence of an imminent overload event.

15.2.4.5.1 Payload Format

The Overload Pause Command SHALL have no payload.

15.2.4.5.2 Effects on Receipt

On receipt of this command, the appliance SHALL pause its operations. In order to resume the normal operation an Overload Pause Resume command SHOULD be issued by the device supporting the client side of the Appliance control cluster.

15.2.4.6 Overload Warning Command

This basic message is used to send warnings the household appliance as a consequence of a possible overload event, or the notification of the end of the warning state.

15.2.4.6.1 Payload Format

The Overload Warning Command payload SHALL be formatted as illustrated in Figure 15-5.

Figure 15-5. Format of the Overload Warning Payload

Octets	2
Data Type	enum8
Field Name	Warning Event

15.2.4.6.2 Payload Details

The Warning Event field represents the identifier of the events that needs to be communicated to the devices to alert about possible overload, as shown in Table 15-7.

Table 15-7. Format of the Event ID Enumerator

Event ID	Description
0x00	Warning 1: overall power above “available power” level
0x01	Warning 2: overall power above “power threshold” level
0x02	Warning 3: overall power back below the “available power” level
0x03	Warning 4: overall power back below the “power threshold” level
0x04	Warning 5: overall power will be potentially above “available power” level if the appliance starts

15.2.4.6.3 Effects on Receipt

On receipt of this command, the appliance SHALL show the possible warning state on a display (e.g., showing an icon with possible overload condition when activating the appliance in case of Warnings 1-2) or resume the normal state in case of events showing the return on normal state (e.g., Warning 3-4).

15.2.5 Server Commands Generated

Table 15-8 lists commands that are generated by the server.

Table 15-8. Cluster-specific Commands Sent by the Server

Command Identifier Field Value	Description	M/O
0x00	Signal State Response	M
0x01	Signal State Notification	M

15.2.5.1 Signal State Response Command

This command SHALL be used to return household appliance status, according to Appliance Status Values and Remote Enable Flags Values.

15.2.5.1.1 Payload Format

The Signal State Response Command payload SHALL be formatted as illustrated in Figure 15-6.

The Appliance Status field: the data field is an 8 bits in length enumerator identifying the appliance status. The enumeration used for this field SHALL match the specifications in Table 15-9.

The Remote Enable Flags and Device Status 2 field: the data field is an 8 bits in length unsigned integer defining remote enable flags and potential appliance status 2 format. The unsigned integer used for this field SHALL match the specifications in Table 15-10.

The Appliance Status 2 field: the command identification is a 24 bits in length unsigned integer representing potential non-standardized or proprietary data.

Figure 15-6. Format of the Signal State Response Command Payload

Octets	1	1	0/3
Data Type	enum8	uint8	uint24
Field Name	Appliance Status	Remote Enable Flags and Device Status 2	Appliance Status 2

15.2.5.1.1.1 Payload Details

ApplianceStatus

ApplianceStatus represents the current status of household appliance. *ApplianceStatus* must be included as part of the minimum data set to be provided by the household appliance device. *ApplianceStatus* is updated continuously as appliance state changes.

Table 15-9 provides states defined.

Table 15-9. Appliance Status Values

Enumeration	Value	Description
OFF	0x01	Appliance in off state
STAND-BY	0x02	Appliance in stand-by
PROGRAMMED	0x03	Appliance already programmed
PROGRAMMED WAITING TO START	0x04	Appliance already programmed and ready to start (e.g., has not reached *StartTime*)
RUNNING	0x05	Appliance is running
PAUSE	0x06	Appliance is in pause
END PROGRAMMED	0x07	Appliance end programmed tasks
FAILURE	0x08	Appliance is in a failure state
PROGRAMME INTERRUPTED	0x09	The appliance programmed tasks have been interrupted
IDLE	0x0a	Appliance in idle state
RINSE HOLD	0x0b	Appliance rinse hold
SERVICE	0x0c	Appliance in service state
SUPERFREEZING	0x0d	Appliance in superfreezing state
SUPERCOOLING	0x0e	Appliance in supercooling state
SUPERHEATING	0x0f	Appliance in superheating state
Manufacturer Specific	0x80..0xff	Manufacturer specific value range

RemoteEnableFlags **Field**

RemoteEnableFlags represents the current status of household appliance correlated with remote control.

RemoteEnableFlags is mandatory and must be included as part of the minimum data set to be provided by the household appliance device.

RemoteEnableFlags is updated continuously when appliance state remote-controllability changes.

Table 15-10 provides details about flags organization.

Table 15-10. Remote Enable Flags Values

Bit Range	Function
0..3	Remote Enable Flags

Bit Range	Function	
	Value	**Enumeration**
	0x0 0x7 0xf 0x1. 0x2..0x06, 0x8..0xe	DISABLED TEMPORARILY LOCKED/DISABLED ENABLED REMOTE CONTROL ENABLED REMOTE AND ENERGY CONTROL Reserved
4..7	Device Status 2 Structure	
	Value	**Enumeration**
	0x0 0x1 0x2 0x3..0xf	PROPRIETARY PROPRIETARY IRIS SYMPTOM CODE Reserved

ApplianceStatus2 Field

ApplianceStatus2 represents a detailed definition of Appliance state. If optionally provided, ApplianceStatus2 is updated continuously as appliance state change.

This field contains non-standardized or proprietary data. In the case of IRIS Symptom Code, 3 bytes representing the 3 digit encoding is provided (possibly complemented with proprietary bytes).

15.2.5.1.2 Effect on Receipt

On receipt of this command, the device is informed of a Household Appliance status.

15.2.5.2 Signal State Notification Command

This command SHALL be used to return household appliance status, automatically when appliance status changes.

15.2.5.2.1 Payload Format

The Signal State Notification Command payload SHALL be formatted as illustrated for the Signal State Response Command Payload.

15.2.5.2.2 Effects on Receipt

On receipt of this command, the device is informed of a Household Appliance status.

15.2.6 Client

The client cluster has no dependencies or specific cluster attributes. The client side of this cluster receives the cluster specific commands generated by the server. The client side of this cluster generates the cluster specific commands received by the server as required by the application.

15.3 EN50523 Appliance Identification

15.3.1 Overview

Please see section 2.2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

Attributes and commands for determining basic information about a device and setting user device information.

The Appliance Identification Cluster is a transposition of EN50523 “Identify Product” functional block.

Note: *Where a physical node supports multiple endpoints it will often be the case that many of these settings will apply to the whole node, that is they are the same for every endpoint on the device. In such cases they can be implemented once for the node, and mapped to each endpoint.*

15.3.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added; CCB 1893

15.3.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	APLNCID	Type 2 (server to client)

15.3.1.3 Cluster Identifiers

Identifier	Name
0x0b00	EN50523 Appliance Identification

15.3.2 Server

15.3.2.1 Attributes

For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed in Table 15-11.

Table 15-11. Appliance Identification Attribute Sets

Attribute Set Identifier	Description
0x000	Basic Appliance Identification
0x001	Extended Appliance Identification

15.3.2.2 Basic Appliance Identification Attribute Set

The Basic Appliance Identification attribute set contains the attributes summarized in Table 15-12.

Table 15-12. Attributes of the Appliance Identification Attribute Set

Identifier	Name	Type	Range	Access	Def	M/O
0x0000	*BasicIdentification*	uint56	-	R	-	M

15.3.2.3 *BasicIdentification* Attribute

BasicIdentification is 56-bit bitmap (7 octets) and contains the basic appliance identification.

BasicIdentification is mandatory and must be included as part of the minimum data set to be provided by the household appliance device.

Table 15-13 provides attribute content specification.

Table 15-13. Basic Appliance Identification Content Specification

Attribute Name	Field	Bits
BasicIdentification	Company ID	0x00-0x0f
	Brand ID	0x10-0x1f
	Product Type ID	0x20-0x2f
	Spec. Ver.	0x37-0x30

Table 15-13 provides Company ID and Brand ID fields content, according to [N2], Table 5.

Table 15-14 provides Product Type IDs field content, again according to [N2] (see Table 6).

Table 15-14. Product Type IDs

Device (Appliance)	Product Type ID
White Goods	0x0000
Dishwasher	0x5601
Tumble Dryer	0x5602

Device (Appliance)	Product Type ID
Washer Dryer	0x5603
Washing Machine	0x5604
Hobs	0x5E03
Induction Hobs	0x5E09
Oven	0x5E01
Electrical Oven	0x5E06
Refrigerator Freezer	0x6601

15.3.2.4 Extended Appliance Identification Attribute Set

The Extended Appliance Identification attribute set contains the attributes summarized in Table 15-15.

Table 15-15. Attributes of the Extended Appliance Identification Attribute Set

Identifier	Name	Type	Range	Acc	Def	M/O
0x0010	*CompanyName*	string	0 to 16 Octets	R	-	O
0x0011	*CompanyId*	uint16	*all*	R	-	O
0x0012	*BrandName*	string	0 to 16 Octets	R	-	O
0x0013	*BrandId*	uint16	*all*	R	-	O
0x0014	*Model*	octstr	0 to 16 Octets	R	-	O
0x0015	*PartNumber*	octstr	0 to 16 Octets	R	-	O
0x0016	*ProductRevision*	octstr	0 to 6 Octets	R	-	O
0x0017	*SoftwareRevision*	octstr	0 to 6 Octets	R	-	O
0x0018	*ProductTypeName*	octstr	2 Octets	R	-	O
0x0019	*ProductTypeId*	uint16	*all*	R	-	O
0x001A	*CECEDSpecificationVersion*	uint8	*all*	R	-	O

15.3.2.5 *CompanyName* Attribute

CompanyName is a ZCL Character String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the UTF-8 format. Example Company Name labels are “Electrolux”, “Indesit Company”, “Candy”. The complete list of valid labels is defined in [E2], Table 7.

15.3.2.6 *CompanyID* Attribute

CompanyID is 16-bit in length unsigned integer which defines the appliance company identifier. The complete list of valid company identifiers is defined in [E2], Table 7.

15.3.2.7 *BrandName* Attribute

BrandName is a ZCL Character String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the UTF-8 format. Example Brand Name labels are “Rex”, “Ariston”, “Hoover”. The complete list of valid labels is defined in [E2], Table 7.

15.3.2.8 *BrandID* Attribute

BrandID is 16-bit in length unsigned integer which defines the appliance brand identifier. The complete list of valid brand identifiers is defined in [E2], Table 7.

Note that Brand Ids and Company Ids are independently defined. The advantage is that one brand of one producer MAY have the same ID as a brand name of another producer.

15.3.2.9 *Model* Attribute

Model is a ZCL Octet String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the UTF-8 format. *Model* defines the appliance model name, decided by manufacturer.

15.3.2.10 *PartNumber* Attribute

PartNumber is a ZCL Octet String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the UTF-8 format. *PartNumber* defines the appliance part number, decided by manufacturer.

15.3.2.11 *ProductRevision* Attribute

ProductRevision is a ZCL Octet String field capable of storing up to 6 character string (the first Octet indicates length) encoded in the UTF-8 format. *ProductRevision* defines the appliance revision code, decided by manufacturer.

15.3.2.12 *SoftwareRevision* Attribute

SoftwareRevision is a ZCL Octet String field capable of storing up to 6 character string (the first Octet indicates length) encoded in the UTF-8 format. *SoftwareRevision* defines the appliance software revision code, decided by manufacturer.

15.3.2.13 *ProductTypeName* Attribute

ProductTypeName is a 2 Octet in length String field which defines the appliance type label. Example *ProductTypeName* labels are “WM”, “RE”, “GO”, respectively for Washing Machine, Refrigerator and Gas Oven. The complete list of valid labels is defined in [E2], Table 8.

15.3.2.14 *ProductTypeID* Attribute

ProductTypeID is a 16-bit in length unsigned integer which defines the appliance type identifier. The structure and complete list of valid *ProductTypeID*s is defined in [E2], Table 7.

15.3.2.15 *CECEDSpecificationVersion* Attribute

CECEDSpecificationVersion is an 8-bit in length unsigned integer which defines the CECED reference documentation. Compliance and certification of appliance communication capabilities can be defined according to Table 15-16 (see [E2], Table 10).

Table 15-16. CECED Specification Version

Specification Version	Value
Compliant with v1.0, not certified	0x10
Compliant with v1.0, certified	0x1A
Compliant with vX.0, not certified	0xX0
Compliant with vX.0, certified	0xXA

15.3.2.16 Commands Received

No cluster-specific commands are received by the server.

15.3.2.17 Commands Generated

No cluster-specific commands are generated by the server.

15.3.3 Client

The client cluster has no dependencies or cluster specific attributes. The client cluster has no cluster specific commands generated or received.

15.4 EN50523 Appliance Events and Alerts

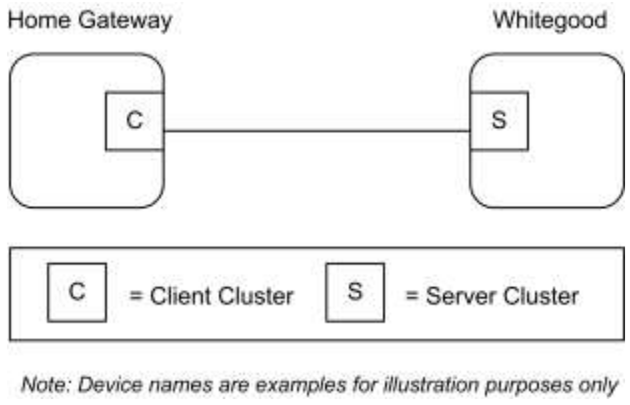
15.4.1 Overview

Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

Attributes and commands for transmitting or notifying the occurrence of an event, such as “temperature reached” and of an alert such as alarm, fault or warning.

It is based on the “Signal event” syntax of EN50523 and completed where necessary.

Figure 15-7. Typical Usage of the Appliance Events and Alerts Cluster


Home Gateway
Whitegood
C
S
C = Client Cluster
S = Server Cluster
Note: Device names are examples for illustration purposes only

There are two different types of occurrences: events and alerts.

Each event is described through two fields:

- An event header
- An event identification value;

The server notifies the client about the event occurred. There is no possibility for the client to get the event from the server and to have a response.

Each alert is described through three fields:

- An alert identification value;
- A category: either WARNING, DANGER, or FAILURE.
- A presence/recovery flag, either the alert has been detected or the alert has been recovered.

The server notifies the client regarding the alerts occurred. The client can also request the alerts from the server and receive the related response.

15.4.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added

15.4.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	APPLEV	Type 2 (server to client)

15.4.1.3 Cluster Identifiers

Identifier	Name
0x0b002	EN50523 Appliance Events and Alerts

15.4.2 Server

15.4.2.1 Attributes

None

15.4.2.2 Commands Received

The received command IDs for the Appliance Events and Alerts Cluster are listed in Table 15-17.

Table 15-17. Received Commands IDs for the Events and Alerts Cluster

Command Identifier Field Value	Description	M/O
0x00	Get Alerts	M

15.4.2.2.1 Get Alerts Command

This basic message is used to retrieve Household Appliance current alerts.

15.4.2.2.1.1 Payload Format

This command does not have a payload.

15.4.2.2.1.2 Effects on Receipt

On receipt of this command, the device SHALL generate a Get Alerts Response command.

15.4.2.3 Commands Generated

The generated command IDs for the Appliance Events and Alerts Cluster are listed in Table 15-18.

Table 15-18. Generated Commands IDs for the Appliance Events and Alerts Cluster

Command Identifier Field Value	Description	M/O
0x00	Get Alerts Response	M
0x01	Alerts Notification	M
0x02	Event Notification	M

15.4.2.3.1 Get Alerts Response Command

This message is used to return household appliance current alerts.

15.4.2.3.1.1 Payload Format

The payload SHALL be formatted as illustrated in Figure 15-8.

Figure 15-8. Format of the Get Alerts Response Command Payload

Octets	1	3	...	3
Data Type	uint8	uint24	...	uint24
Field Name	Alerts Count[164]	Alert structure 1	...	Alert structure *n*

15.4.2.3.1.1.1 Payload Details

The **Alerts Count** field: the data field is an 8 bits in length unsigned integer, containing the following alerts structures count and alert structure type.

Table 15-19 provides details about Alerts Count and Structure field organization.

Table 15-19. Alert Count Organization

Bit range	**Function**	
0..3	Number of Alerts n	
4..7	Type of alert	
	Value	**Enumeration**
	0x0 0x1..0xf	UNSTRUCTURED Reserved

[164] Even if the ApplianceAlertList array number of element field is 16-bit in length, the actual content is limited to 0x000**n**, where, in actual implementations, n is lower than 255 (except for the invalid condition, 0xffff). Then, the notification of the Alert count is mapped to a single byte (following appliance interworking specifications).

Each *Alerts Structure* field SHALL be formatted as illustrated in Table 15-20.

Table 15-20. Alerts Structure Organization

Bit range	Function	
0..7	Alert id	
8..11	Category	
	Value	**Enumeration**
	0x0 0x1 0x2 0x3 0x4 – 0xf	Reserved WARNING DANGER FAILURE Reserved
12..13	Presence recovery	
	Value	**Enumeration**
	0x0 0x1 0x2 – 0x3	RECOVERY PRESENCE Reserved
16..23	Manufacturer specific bits	

The *Alert ID* field can have the following values:

- Value 0 is reserved.
- Values ranging from 1 to 63 are standardized.
- Values ranging from 64 to 127 are reserved.
- Values ranging from 128 to 255 are manufacturer specific.

15.4.2.3.1.2 Effects on Receipt

On receipt of this command, the device is informed of a Household Appliance warning and fault occurrence.

15.4.2.3.2 Alerts Notification Command

This message is used to notify the current modification of warning and/or fault conditions.

15.4.2.3.2.1 Payload Format

The payload SHALL be formatted as illustrated in Figure 15-9.

Figure 15-9. Format of the Alerts Notification Command Payload

Octets	1	3	...	3
Data Type	uint8	uint24	...	uint24
Field Name	Alerts Count	Alert structure 1	...	Alert structure *n*

15.4.2.3.2.1.1 Payload Details

See Get Alert Response command.

15.4.2.3.2.2 Effects on Receipt

On receipt of this command, the device is informed of a Household Appliance warning and fault occurrence.

15.4.2.3.3 Event Notification Command

This message is used to notify an event occurred during the normal working of the appliance.

15.4.2.3.3.1 Payload Format

The payload SHALL be formatted as illustrated in Figure 15-10.

Figure 15-10. Format of the Event Notification Command Payload

Octets	1	1
Data Type	uint8	uint8
Field Name	Event Header	Event Identification

15.4.2.3.3.1.1 Payload Details

The *Event Header* is a reserved field set to 0.

The *Event Identification* field: the *Event Identification* is an 8-bits in length field identifying the event to be notified. The codes used for this field SHALL match those shown in Table 15-21:

Table 15-21. Event Identification

Event Identification	**Value**	**Description**
END_OF_CYCLE	0x01	End of the working cycle reached
TEMPERATURE_REACHED	0x04	Set Temperature Reached
END_OF_COOKING	0x05	End of cooking process
SWITCHING OFF	0x06	
Manufacturer Specific	0x40-0xf6	Manufacturer specific Id range
WRONG_DATA	0xf7	

Event Identification	**Value**	**Description**
Manufacturer Specific	0xf8-0xff	Manufacturer specific Id range

15.4.2.3.3.2 Effects on Receipt

On receipt of this command, the device is informed of a Household Appliance working event occurrence.

15.4.3 Client

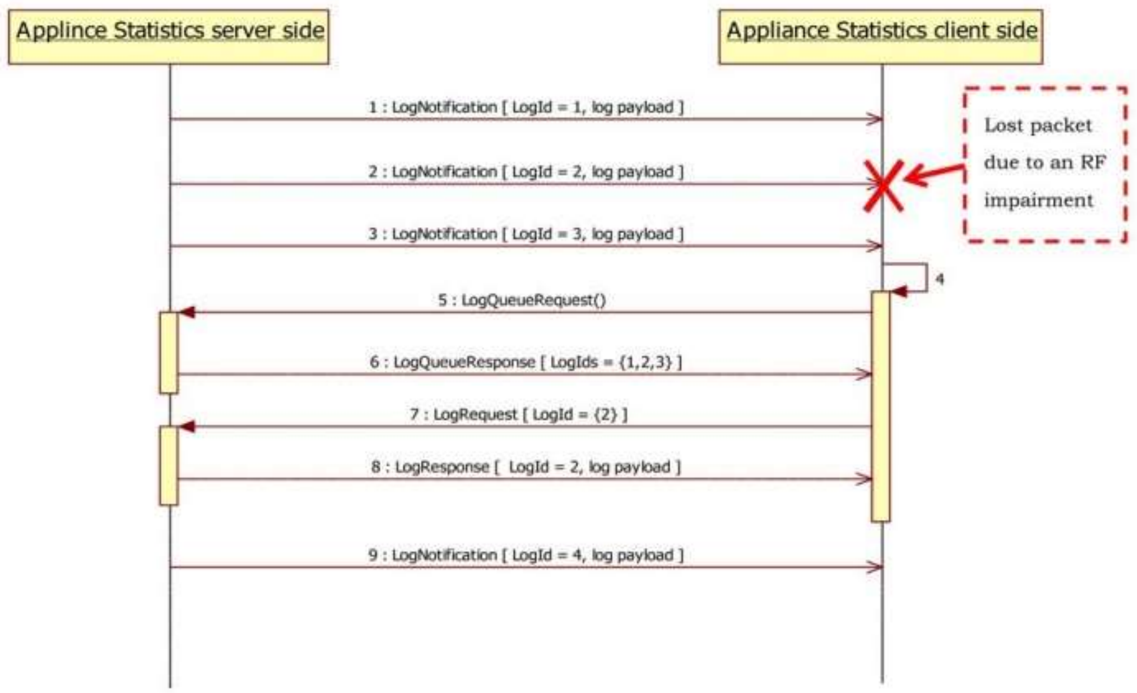
The client cluster has no dependencies or specific cluster attributes. The client side of this cluster receives the cluster specific commands generated by the server. The client side of this cluster generates the cluster specific commands received by the server as required by the application.

15.5 Appliance Statistics

15.5.1 Overview

This cluster provides a mechanism for the transmitting appliance statistics to a collection unit (gateway). The statistics can be in format of data logs. In case of statistic information that will not fit the single payload, the Partition cluster SHOULD be used.

Each appliance uses persistent memory to temporarily store collected statistic logs (entries). The maximum number of stored statistic logs is appliance dependent. If some log notification packets are lost due to temporary unreliable RF communication, sequential Log IDs allow the collection of the missing logs. The following is a simple example of an application-level policy used for log collection. When receiving logs with non-consecutive Log IDs, the client can ask server side for the available log queue to verify the actual availability of the missing log. If present, the log can be explicitly retrieved using the LogRequest command.


Applince Statistics server side
Appliance Statistics client side
1 : LogNotification [LogId = 1, log payload]
2 : LogNotification [LogId = 2, log payload]
Lost packet due to an RF impairment
3 : LogNotification [LogId = 3, log payload]
4
5 : LogQueueRequest()
6 : LogQueueResponse [LogIds = {1,2,3}]
7 : LogRequest [LogId = {2}]
8 : LogResponse [LogId = 2, log payload]
9 : LogNotification [LogId = 4, log payload]

15.5.1.1 Revision History

Rev	Description
1	mandatory global *ClusterRevision* attribute added;CCB 1893

15.5.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	APPLST	Type 2 (server to client)

15.5.1.3 Cluster Identifiers

Identifier	Name
0x0b003	EN50523 Appliance Statistics

15.5.2 Server

15.5.2.1 Attributes

The server side of this cluster contains the attributes the statistics and log information shown in Table 15-22.

Table 15-22. Server Attributes

Identifier	Description	Type	Access	Default	M/O
0x0000	*LogMaxSize*	uint32	R	0x0000003C	M
0x0001	*LogQueueMaxSize*	uint8	R	0x01	M

15.5.2.1.1 *LogMaxSize* Attribute

The *LogMaxSize* attribute describes the maximum size of a log payload that can be transferred using the Log Notification and Log Response commands. In case the *LogMaxSize* attribute is greater than 70 bytes (0x46) the Appliance Statistics commands SHOULD be transferred using the partition cluster. This is the case of a “bulk log” transferred from a server side (e.g., White Goods) to a client side (e.g., home gateway) of the Appliance Statistics Cluster.

15.5.2.1.2 *LogQueueMaxSize* Attribute

The *LogQueueMaxSize* attribute describes the maximum number of logs that are available in the server side of the Appliance Statistics cluster. The logs MAY be retrieved by the client using the Log Request command.

15.5.2.2 Commands

The generated command IDs for the Appliance Statistics Server are listed in Table 15-23.

Table 15-23. Commands Generated by the Appliance Statistics Server

Command ID	Description	M/O
0x00	Log Notification	M
0x01	Log Response	M
0x02	Log Queue Response	M
0x03	Statistics Available	M

15.5.2.2.1 Log Notification

The Appliance Statistics Cluster server occasionally sends out a Log Notification command to the devices to which it needs to log information related to statistics (e.g., home gateways) which implement the client side of Appliance Statistics Cluster.

15.5.2.2.1.1 Payload Format

Figure 15-11. Format of the Log Notification Payload

Octets	4	4	4	1	…	1
Data Type	UTC	uint32	uint32	data8	data8	data8
Field Name	Time Stamp	Log ID	Log Length	Log Payload		

15.5.2.2.1.2 When Generated

The Log Notification command is generated when the appliance needs to send log information related to its statistics to a remote device (e.g., home gateway) without being solicited by the client side. The log information sent with the Log Notification command from the server side is not solicited by specific command generated by the client side of the Appliance Statistics cluster. The Log ID field identifies uniquely the log information contained in the log payload. Log IDs SHALL be consecutive. Log Length field indicated the length in bytes of the log payload and SHALL be less than *LogMaxSize* attribute.

If the device generating the Log Notification command is not able to generate the time stamp information it SHALL insert an invalid UTC Time (0xffffffff). In this case the server side of the Appliance statistics cluster (e.g., a home gateway) SHOULD insert a timestamp of the received log notification if available before storing or transmitting the log information to backend systems.

15.5.2.2.1.3 Effect Upon Receipt

Upon receipt of the Log Notification command, the Appliance statistics client will respond with a Default Response command if requested or if an error occurs. In case of error the server side of Appliance statistics cluster MAY store the information in the queue and notify the client that there are statistics available by using the Statistic Available command.

15.5.2.2.2 Log Response

The Appliance Statistics Cluster server sends out a Log Response command to respond to a Log Request command generated by the client side of the Appliance Statistics cluster.

15.5.2.2.2.1 Payload Format

The payload of the Log Response command is the same as the Log Notification command.

15.5.2.2.2.2 When Generated

The Log Response command is generated to respond to Log Request sent from a device supporting the client side of the Appliance Statistics cluster (e.g., home gateway).

15.5.2.2.2.3 Effect Upon Receipt

Upon receipt of the Log Response command, the Appliance statistics client will respond with a Default Response command if requested or if an error occurs.

15.5.2.2.3 Log Queue Response

The Log Queue Response command is generated as a response to a Log Queue Request command in order to notify the client side of the Appliance statistics cluster about the logs stored in the server side (queue) that can be retrieved by the client side of this cluster through a Log Request command. Please note that the LogQueueSize field SHALL be less than the *LogQueueMaxSize* attribute.

15.5.2.2.3.1 Payload Format

Figure 15-12. Format of the Log Queue Response Payload

Octets	1	4	4	4
Data Type	uint8	uint32	…	uint32
Field Name	Log Queue Size	Log ID	…	Log ID

15.5.2.2.3.2 When Generated

The Log Queue Response command is generated in response to a Log Queue Request sent from a device supporting the client side of the Appliance Statistics cluster (e.g., home gateway) Please note that if Log Queue Size is equal to zero (not logs in the queue), the packet SHALL not carry Log IDs.

15.5.2.2.3.3 Effect Upon Receipt

Upon receipt of the Log Queue Response command, the Appliance statistics client will respond with a Default Response command if requested or if an error occurs. The client side of the appliance statistics willing to get the logs in the queue SHALL then use only the Log IDs that have been indicated in the Log Queue Response.

15.5.2.2.4 Statistics Available

The Appliance Statistics Cluster server sends out a Statistic Available command to notify the client side of the Appliance Statistics cluster that there are statistics that can be retrieved by using the Log Request command.

15.5.2.2.4.1 Payload Format

The Statistic Available command is the same as the Log Queue Response command. The Log IDs that can be retrieved by the client are indicated in the payload.

15.5.2.2.4.2 When Generated

The Statistic Available command is generated to notify a device supporting the client side of the Appliance Statistics cluster (e.g., home gateway) to get the statistics information from the log queue as soon as available to perform this operation.

15.5.2.2.4.3 Effect Upon Receipt

Upon receipt of the Statistic Available command, the client side of the Appliance Statistics cluster is notified on the availability of statistics in the server side that can be retrieved by using Log Request commands.

The Appliance statistics client will respond with a Default Response command if requested or if an error occurs.

15.5.3 Client

15.5.3.1 Attributes

There are no attributes on the client side of the Appliance Statistics Cluster.

15.5.3.2 Commands

The generated command IDs for the Appliance Statistics Client are listed in Table 15-24.

Table 15-24. Commands Generated by the Appliance Statistics Client

Command ID	Description	M/O
0x00	Log Request	M
0x01	Log Queue Request	M

15.5.3.2.1 Log Request

The Log Request command is send from a device supporting the client side of the Appliance Statistics cluster (e.g., Home Gateway) to retrieve the log from the device supporting the server side (e.g., appliance).

15.5.3.2.1.1 Payload Format

Figure 15-13. Format of the Log Request Payload

Octets	4
Data Type	uint32
Field Name	Log ID

15.5.3.2.1.2 When Generated

The Log Request command is generated to retrieve a log information from a device supporting the server side of the Appliance Statistics cluster (e.g., appliance). The log information is addressed by referencing it with the Log ID field. In order to get the Log ID that can be retrieved with the Log Request command, the Log Queue Request command MAY be used.

15.5.3.2.1.3 Effect Upon Receipt

Upon receipt of the Log Request command, the Appliance statistics server will respond with a Log Response command if the log is available or with a Default Response if an error occurs. In case the Log ID is not available in the server side of the cluster the status code carried by the Default Response SHALL be “NOT_FOUND.”

15.5.3.2.2 Log Queue Request

The Log Queue Request command is sent from a device supporting the client side of the Appliance Statistics cluster (e.g., Home Gateway) to retrieve the information about the logs inserted in the queue, from the device supporting the server side (e.g., appliance).

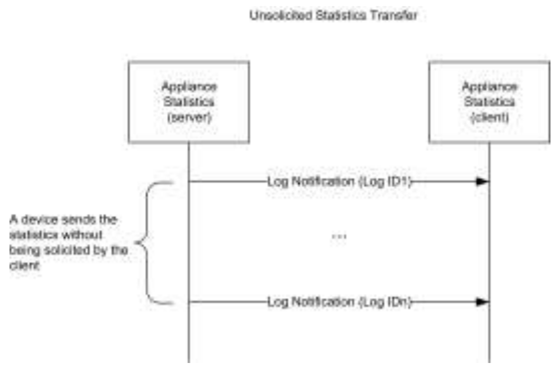
15.5.3.2.2.1 Payload Format

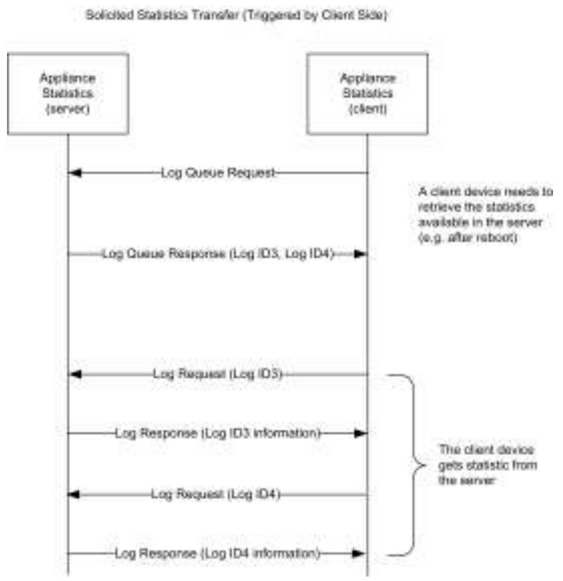
The Log Queue Request command has no payload.

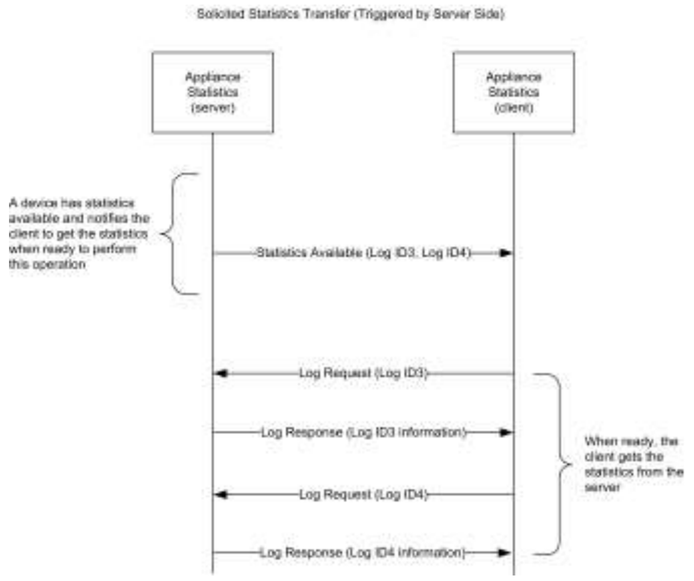
15.5.4 Appliance Statistics Cluster Sequence Diagram

Figure 15-14 shows a typical sequence interaction between the client and server sides of the Appliance Statistics Cluster.

Figure 15-14. Appliance Statistics Cluster Sequence Diagram


Unsolicited Statistics Transfer
Appliance Statistics (server)
Appliance Statistics (client)
Log Notification (Log ID1)
A device sends the statistics without being solicited by the client
...
Log Notification (Log IDn)


Solicited Statistics Transfer (Triggered by Client Side)
Appliance Statistics (server)
Appliance Statistics (client)
Log Queue Request
A client device needs to retrieve the statistics available in the server (e.g. after reboot)
Log Queue Response (Log ID3, Log ID4)
Log Request (Log ID3)
Log Response (Log ID3 information)
The client device gets statistic from the server
Log Request (Log ID4)
Log Response (Log ID4 information)


Solicited Statistics Transfer (Triggered by Server Side)
Appliance Statistics (server)
Appliance Statistics (client)
A device has statistics available and notifies the client to get the statistics when ready to perform this operation
Statistics Available (Log ID3, Log ID4)
Log Request (Log ID3)
Log Response (Log ID3 information)
When ready, the client gets the statistics from the server
Log Request (Log ID4)
Log Response (Log ID4 information)