ZIGBEE SMART ENERGY STANDARD

ZigBee Smart Energy Standard

ZigBee Profile: 0x0109

Revision 18

Version 1.1b

ZigBee Document 07-5356-18	
November 17, 2012 2:35 am	
Sponsored by: ZigBee Alliance	
Accepted by	ZigBee Alliance Board of Directors.
Abstract	
Keywords	ZigBee, Profile, AMI, Application Framework, Smart Energy, Standard.

November 17, 2012

This page intentionally blank

NOTICE OF USE AND DISCLOSURE

ZigBee Alliance, Inc.
2400 Camino Ramon, Suite 375
San Ramon, CA 94583, USA

This page intentionally blank

TABLE OF CONTENTS

This page intentionally blank

LIST OF TABLES

LIST OF FIGURES

PARTICIPANTS

Contact Information

Much of the information in this document is preliminary and subject to change. Members of the ZigBee Working Group are encouraged to review and provide inputs for this proposal. For document status updates, please contact:

Phil Jamieson
Philips, Cross Oak Lane,
Redhill, Surrey, RH1 5HA, UK

E-Mail: phil.jamieson@philips.com
Phone: +44 1293 815265
Fax: +44 1293 815050

You can also submit comments using the ZigBee Alliance reflector. Its web site address is:

www.zigbee.org

The information on this page should be removed when version 0.9 is accepted by the Working Group.

Participants

The following is a list of those who were members of the ZigBee Architecture Review Committee (ZARC) leadership when this document was released:

Skip Ashton: Chair

Phil Jamieson: Vice-Chair

When the document was released, the Smart Energy Application Profile Work Group leadership was composed of the following members:

Larry Kohrmann: Chair

Ian Winterburn: Vice-Chair

Rob Alexander & David Smith: Technical Editors

Jeff Shudark: Secretary

Contributions were made to this document from the following members:

Rob Alexander	Tim Enwall	John Knuth	Yuri Shteinman
Skip Ashton	Tony Field	Zin Kyaw	Robby Simpson
Wally Barnum	Chris Fitzer	Christopher Leidigh	Sumit Singh
Gary Birk	Benjamin Gartner	Bill Lichtensteiger	David Smith
Rick Bojahra	Daniel Gavelle	John Lin	Matt Smith
Mark Borins	Dean van Gerrevink	Dan Lohman	Zachary Smith
John Buffington	Tim Gillman	Juan Agüí Martín	Michael G. Stuber
Jeff Cooper	Jesper Hæe	Tom Martin	Don Sturek
Damon Corbin	Matt Hamblin	Jeff Mathews	Jakob Thomsen
Michael Cowan	Jim Hartman	Tony Mauro	Jesper Thomsen
John Cowburn	Donald Hasson	Luca Negri	Michel Veillette
Robert Cragie	Heinz Hohl	Howard Ng	Mads Westergreen
Jonathan Cressman	Jeff Hugghins	Eugene Peng	Ian Winterburn
Jeff Drake	Rolf Kistler	John Prater	Jack Worsnop

DOCUMENT HISTORY

Table 1.1 shows the change history for this specification.

Table 1.1 Document Revision Change History

Revision	Version	Description
0		Original version.
1		First draft to include annexes and cluster information.
2		Updated to include Key Establishment Cluster Annex. Added other minor changes within the document.
3		Included comments from internal Smart Energy (formerly Smart Energy) group review. New Items: Added Power Factor in Simple Metering cluster. Added Group support in the DR/LC cluster.
4		Included comments from internal Smart Energy group review scattered throughout. A number of field and attribute adds to the clusters.
5		Corrected Document Number issues, otherwise same as Revision 4.
6		Additional changes: Usual grammar and spelling changes *Load Profile* commands have been updated. Added Attributes to support the latest partial LP interval. Load Control rules for DR/LC Randomization ESP Historical Attributes. changes in Simple Metering cluster Changes to the *Get Current Price* command
7		Grammar, spelling, and formatting changes
8		PDF version of 07
9		First pass at comment resolution. Please refer to document #075424r03ZB for changes.
10		Second pass at comment resolution. Please refer to document #075424r04ZB for changes. Renamed to Smart Energy Profile.

Table 1.1 Document Revision Change History (Continued)

11		Third pass at comment resolution. Please refer to document #075424r05ZB for changes. Moved SE Cluster definitions into the annex D. Significant changes in the Security related sections. Best practice information added to more sections. Updated Annex E covering overlapping event examples. Corrected issues relating to the following CCBs: CC-900 [SE] Randomizing Price Events CC-901 [SE] Message Cluster Start Time CC-902 [SE] New Status Field for *Get Profile Response* Command CC-903 [SE] Support for Binding CC-904 [SE] ESP Historical Consumption Attributes in Simple Metering Server CC-905 [SE] More Precise Event Status Enumeration for *Report Event Status* Command CC-906 [SE] Additional Description of Device Class bits 0 and 1 CC-907 [SE] Array vs. Series of Intervals for *Get Profile Response* Command CC-908 [SE] Randomization of *Report Event Status* Command Send Times CC-909 [SE] Effective Time Field of Cancel Load Control Event to be Mandatory CC-910 [SE] Consolidation of Joining Procedures CC-911 [SE] Out of Bands Methods of Authentication CC-912 [SE] Method to Make Registered Devices Listed on ESP CC-913 [SE] Clarification of Rate Label Field of *Publish Price* Command.
12		PDF version of 075356r11.
13		Converted from Word to FrameMaker, includes all CCBs called out in the SE Profile Errata 08119r08.
14		Final editorial changes for initial publication.

Table 1.1 Document Revision Change History (Continued)

15		Corrected issues related to the following CCBs (from errata document 084914r05): CC-964 ZigBee Cluster Library reference doesn't contain the revision number. CC-965 Specification needs to clarify the service discovery process steps prior to and after the Key Establishment process. End Devices must also initiate the processes. CC-966 The Identify cluster should be Optional, not mandatory. CC-967 The Common Features and Functions table incorrectly calls out the binding and service discovery requests as mandatory items. CC-968 Future definitions of fields added to the end of commands are to be treated as reserved fields. CC-973 Addition of Greenhouse Gas (CO_2) pricing information to the *Publish Price* command. CC-974 Addition of Supply Limit tracking in the Metering[a] Cluster. CC-980 Correct and describe CRC Algorithm used for Installation Codes. CC-981 Correct the Installation Codes text examples and provide example source code for testing/using the MMO Hash Algorithm. CC-982 Attributes *CurrentPartialProfileIntervalValueDelievered* and *CurrentPartialProfileIntervalValueReceived* do not list default values or mandatory/optional status. CC-983 Attributes *Power Factor*, *ReadingSnapShotTime*, *CurrentMaxDemandDelieveredTime*, and *CurrentMaxDemandReceivedTime* are incorrectly replicated in another section. Corrected issues related to the following CCBs: CC-984 Addition of Key Establishment test vectors. CC-986 Addition of metering device types to the simple metering cluster attribute *MeteringDeviceType Enumeration*. CC-993 Initiate Key Establishment Request and Response Payload Format field names need to match field names defined in Payload Format figures.

Table 1.1 Document Revision Change History (Continued)

16		CC-923 Best Practices for Client devices using the Inter-PAN Transmission section (Annex B) CC-940 Rename Simple Metering to just Metering CC-996 Test & Profile Specification Conflict (Message Confirmation) CC-1002 Typos (*Publish Price* Command Start Time description.) CC-1015-ESP Historical Consumption CC-1018 Mirror Device CC-1026 Remove term “unsecure rejoin” from document. CC-1027 Messaging cluster message payload size CC-1028 Add DRLC Commands Received to D.2.2 CC-1030 Range Extender does not allow support of optional clusters. CC-1031 Price Cluster client server references transposed CC-1032 Price Server Cluster Attributes CC-1059 Extra word CC-1060 Rijndael source code URL no longer valid CC-1069 Update reference to ZCL specification (075123r02) CC-1070 Publish Price payload format clarification and value of unused optional fields. CC-1072 ZigBee Smart Energy naming of ESI CC-1077 Interval Channel data type CC-1082 Range of InstantaneousDemand CC-1083 Price cluster clarifications CC-1087 Price clients cannot request all price values CC-1090 Example string is too long CC-1096 Message Confirmation Payload Details Typo CC-1098 Key establishment confirm key response CC-1103 Demand limit enabled or not CC-1108 Allow more flexibility with Issuer Event ID CC-1118 Recommended Practices for devices in a multi-ESP HAN CC-1119 Responding to a Get message when server's list is empty CC-1124 Add mcf Unit of Measure

Table 1.1 Document Revision Change History (Continued)

16		CC-1125 Clarify the text describing Issuer field and known CA CC-1130 Binding Clarification CC-1135 Change Attribute Access CC-1159 CMU Resolve CMU Audit response editorial comments CC-1160 Resolve CMU Audit response technical comments CC-1170 Add unitless Unit of Measure CC-1173 Duty Cycle Proposal CC-1179 Alarms Cluster support in Smart Energy Metering Device CC-1180 Addition of MJ (Mega Joule) Incremental Release 1 CCB 1181 Addition of optional attributes to the metering cluster required for district heat and cooling metering CCB 1195 Signature on DRLC messages are unused and should not be required CCB 1198 Data Type of MeteringDeviceType CCB 1206 When is the Price Acknowledegment Generated? CCB 1207 Is Price Acknowledgement optional? CCB 1210 Optional Alternate Cost Attribute do not specify unused defaults CCB 1244 Meaning of Start Time in Get Scheduled Events is ambiguous CCB 1320 Final SE 1.1 Interop Issues CCB Event ID backwards compatibility issue[b]

Table 1.1 Document Revision Change History (Continued)

17		CCB 994 - InterPAN Messaging should not allow any message CCB 1217 - Clarification on access control for Mirroring CCB 1218 - Prepayment cluster should also be mirrored CCB 1219 - Add push (report attribute) to align with Mirroring section later on CCB 1226 - Naming of SE Display Device CCB 1241 - Trust Center Swap-Out not explicitly optional CCB 1243 - Adjusting events with Start Time = 0x00000000 CCB 1258 - Need modifications to SE 1.1 specification for OTA CCB 1262 - Mirror CCB 1264 - Add support for CV and PTZ (gas conversion factors) to Price cluster CCB 1265 - Improvements to the Handling of Multiple Fuels CCB 1267 - Change End Device restriction from a SHALL to a SHOULD CCB 1268 - Price Tier Sub-fields are not sequential CCB 1269 - Multiplier and Divisor attributes should also apply to newly added EnergyCarrier and Temperature attributes CCB 1270 - Temperature attributes ought to be signed 24 bit integers CCB 1273 - Addition of 'Get Supported Tunnel Protocols' Command and Response CCB 1284 - Start-up Parameter TC Address allows non-Coordinator CCB 1289 - PhysicalEnvironment bit for Mirroring CCB 1292 - OctetString payload octet counts incorrect CCB 1293 - Start/Stop randomization for DRLC and Price CCB 1294 - Miscellaneous editorial comments CCB 1300 - ThreshholdMultiplier behavior when 0 needs definition CCB 1322 - Inconsistent spelling of Enrolment CCB 1324 - SignatureType and Signature not marked Optional (O) in Figure D.5 CCB 1332 - Price Tier CCB 1334 - "Publish Price" typo as "Public Price" command CCB 1339 - Event Override CCB 1341 - kW and kWh in table D.22 CCB 1347 - Clarification on Publish Price command

Table 1.1 Document Revision Change History (Continued)

17		CCB 1349 - Time client should be allowed on ESI CCB 1350 - Add recommended practice for time synchronizing ESIs CCB 1352 - Modification of Multi-ESI mechanism from inter-op event CCB 1353 - Tunneling cluster transfer size establishment is incomplete CCB 1355 - Tunneling cluster CloseTimeout attribute should have special behavior when set to 0 CCB 1376 - Metering Device Types CCB 1380 - Unclear what happens when a received LCE event is "ignored" CCB 1382 - Typo in text CCB 1383 - Commodity Type to be read after service discovery to understand type of Price server CCB 1384 - Wrong data type for Supply Status CCB 1389 - Event ID backwards compatibility issue CCB 1397 - Clarify cluster usage of security keys CCB 1398 - Add a Signature Type of None CCB 1401 - Add TunnelClosureNotification command CCB 1403 - Permit Join Best Practices CCB 1404 - TC behavior unclear when devices leave network CCB 1419 - Trust Center Swap-out - Bindings & Mirrors CCB 1437 - DeviceClass is marked read-writeable but write may not be allowed CCB 1440 - Mirroring Feature CCB 1452 - Visibility of Mirror Endpoints CCB 1486 - End point requirement
18	1.1b	CCB 1275 - Extended PAN ID CCB 1276 - CCB 996 CCB 1283 - IPD required clusters draft text CCB 1285 - EUI64 mandatory in TC swap-out procedure CCB 1286 - Conflicting requirements for join behaviors

Table 1.1 Document Revision Change History (Continued)

18	1.1b	CCB 1316 - DeviceClass Value Type/Range CCB 1318 - DRLC Server Does Not Specify Minimum Number of Events CCB 1325 - Number of Events incorrectly references GetBlockPeriod command CCB 1333 - Range of Price Ratio, Generation Price Ratio CCB 1346 - Heating/Cooling Set Point Ranges CCB 1348 - Response to Cancel Load Control Event CCB 1441 - Average Load Adjustment Percentage Text Appears to be Incorrect CCB 1449 - Actions Taken before Time Synced CCB 1455 - DRLC Cancel Load Control - Ignore Effective Time CCB 1456 - DRLC Cancel Load Control with different filters than the one creating the event. CCB 1457 - IHDs shall have the Device Types of the DRLC Events they wish to Display CCB 1482 - Conflicting Behavior for TC Swap-out and Rejoin when using Installation Codes CCB 1491 - Trust Center Keep Alive messages must be APS encrypted CCB 1494 - Add Billing Period Attribute Set to the Price Cluster [RIB] CCB 1500 - New Metering Attribute for Block Pricing [RIB] CCB 1537 - Previous Changes added Requirements to Incorrect Section [RIB] CCB 1547 - Consolidate Block Pricing Requirements [RIB] CCB 1564 - Unclear on the IPD cluster requirement CCB 1570 - Rename Display Device to "In-Home Display" (IHD) CCB 1572 - Missing Document History CCB 1592 - Missing word in sentence makes spec confusing Minor typographical corrections resulting from the document ballot

a. CCB 940

b. CCB 1572

CHAPTER

1

INTRODUCTION

1.1 Scope

This Standard defines device descriptions and standard practices for Demand Response and Load Management "Smart Energy" applications needed in a Smart Energy based residential or light commercial environment. Installation scenarios range from a single home to an entire apartment complex. The key application domains included in this initial version are metering, pricing and demand response and load control applications. Other applications will be added in future versions.

1.2 Purpose

This specification provides standard interfaces and device definitions to allow interoperability among ZigBee devices produced by various manufacturers of electrical equipment, meters, and Smart Energy enabling products.

1.3 Provisional Features

Some of the features in this version of this specification are provisional and non-certifiable. The text regarding these features may change before reaching certifiable status. The features consist of the following items:

- Metering cluster Historical Consumption attributes 0x09-0x0E, 0x11 and 0x12
- Metering cluster Alarms attribute set.
- Price cluster Commodity attribute set, with the exception of the *Standing Charge* attribute.
- Price cluster Price Tier values above 0x06.

- Price cluster *Get Block Price* command.
- Price cluster *Publish Block Period* command.
- The Price cluster client attributes.
- Tunneling cluster Flow Control option.
- Prepayment cluster.
- Trust Center Swapout behaviors.

CHAPTER

2

REFERENCES

2.1 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.

2.1.1 ZigBee Alliance Documents

[B1] ZigBee document 07-5123-04[1], ZigBee Cluster Library Specification, ZigBee Cluster Library Development Board.

[B2] ZigBee document 064309r04, Commissioning Framework

[B3] ZigBee Document 053474r18[2], The ZigBee Specification, ZigBee Technical Steering Committee (TSC)

[B4] ZigBee Document 03084r00, ZigBee Key Establishment Proposal Certicom

[B5] ZigBee 075297r04, Proposal for Inter-PAN Exchange of Data in ZigBee

[B6] ZigBee document 095343r01, Installation Code Sample Source Code[3]

[B7] ZigBee document 08006r03, ZigBee 2007 Layer PICS and Stack Profiles, ZigBee Core Stack Working Group

1. Incremental Release 1
2. Incremental Release 1
3. CCB 1060

[B8] Over the Air Upgrade Cluster Spec. 09-5264-19

[B9] Over the Air Upgrade Cluster test spec: 09-5473-06

[B10] Over the Air Upgrade Cluster PICs: 09-5284-09

2.1.2 External Reference Documents

[B11] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.15.4 2003, IEEE Standard for Information Technology Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). New York: IEEE Press. 2003

[B12] ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), American Bankers Association. Available from http://www.ansi.org.

[B13] ANSI X9.63-2001, Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography, American Bankers Association, November 20, 2001. Available from http://www.ansi.org.

[B14] NIST Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised), March 2007. Available from http://csrc.nist.gov.

[B15] NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, May 2004. Available from http://csrc.nist.gov.

[B16] FIPS Pub 197, Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/N.I.S.T, Springfield, Virginia, November 26, 2001. Available from http://csrc.nist.gov.

[B17] FIPS Pub 198, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198, US Department of Commerce/N.I.S.T., Springfield, Virginia, March 6, 2002. Available from http://csrc.nist.gov.

[B18] Standards for Efficient Cryptography: SEC 1 (working draft) ver 1.7: Elliptic Curve Cryptography, Certicom Research, November 13, 2006. Available from http://www.secg.org

[B19] Standards for Efficient Cryptography: SEC 4 (draft) ver 1.1r1: Elliptic Curve Cryptography, Certicom Research, June 9, 2006. Available from http://www.secg.org

[B20] RFC 3280: Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile. IETF, April 2002. Available from http://www.ietf.org

[B21] Standards for Efficient Cryptography: SEC 4 (draft) ver 1.1r1: Elliptic Curve Cryptography, Certicom Research, June 9, 2006. Available from http://www.secg.org

[B22] RFC 3280: Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile. IETF, April 2002. Available from http://www.ietf.org

This page intentionally blank

CHAPTER

3

DEFINITIONS

3.1 Conformance Levels

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

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

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

Should: A key word indicating that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; that a certain course of action is preferred but not necessarily required; or, that (in the negative form) a certain course of action is deprecated but not prohibited (“should” equals “is recommended that”).

3.2 ZigBee Definitions

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

Cluster: A container for one or more attributes and/or messages in a command structure.

Cluster identifier: A reference to the unique enumeration of clusters within a specific application profile. The cluster identifier is a 16-bit number unique within the scope of the application profile and identifies a specific cluster. Cluster identifiers are designated as inputs or outputs in the simple descriptor for use in creating a binding table.

Device: A description of a specific device within an application profile. For example, the light sensor device description is a member of the home automation application profile. The device description also has a unique identifier that is exchanged as part of the discovery process.

Node: Same as a unit.

Product: A product is a unit that is intended to be marketed. It implements application profiles that may be a combination of private, published, and standard.

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

Unit: A unit consists of one or more physical objects (e.g., switch, controller, etc.) and their corresponding application profile(s) that share a single 802.15.4 radio. Each unit has a unique 64-bit IEEE address.

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.

CHAPTER

4

ACRONYMS AND ABBREVIATIONS

AES	Advanced Encryption Standard
AMI	Advanced Metering Infrastructure or Advanced Metering Initiative
BPL	Broadband over Power Lines
CA	Certificate Authority
CBKE	Certificate-based Key Establishment
CT	Commissioning Tool
ECDSA	Elliptic Curve Digital Signature Algorithm
ECMQV	Elliptic Curve Menezes-Qu-Vanstone
EMS	Energy Management System
EPID	Extended PAN Identifier
ESI	Energy Service Interface[a]
EUI64	Extended Universal Identifier-64
GPRS	General Packet Radio Service
HA	Home Automation
HAN	Home Area Network
IHD	In-Home Display
IPD	In-Premises Display (Same as IHD)[b] or Inter-PAN Device[c]
IVR	Interactive Voice Response
MAC	Medium Access Control (referring to protocol stack sublayer)
MAC	Message Authentication Code (referring to cryptographic operation)
MRD	Market Requirements Document
NAN	Neighborhood Area Network

PAN	Personal Area Network
PKKE	Public Key Key Establishment
PCT	Programmable Communicating Thermostat
PID	PAN Identifier
RFD	Reduced Functionality Device
SAS	Startup Attribute Set
SE	Smart Energy
SKKE	Symmetric Key Key Exchange
TC	Trust Center
TOU	Time of Use
UKE	Unprotected Key Establishment
UTF-8	8-bit Unicode Transformation Format Unicode Transformation Format
ZCL	ZigBee Cluster Library
ZDP	ZigBee Device Profile

a. CCB 1072

b. CCB 1226

c. CCB 1570

CHAPTER

5

PROFILE DESCRIPTION

5.1 A ZigBee Smart Energy Network

The Smart Energy market requires two types of ZigBee networks for metering and energy management. These include neighborhood area networks for meters, using ZigBee for sub-metering within a home or apartment, and using ZigBee to communicate to devices within the home. Different installations and utility preferences will result in different network topologies and operation and this profile must allow for these differences. However, each of these networks will operate using the same Basic Principles to ensure interoperability.

Because of the type of data and control within the Smart Energy network, application security is a key requirement. The application will use link keys which are optional in the ZigBee and ZigBee Pro stack profiles but are required within a Smart Energy network. The Trust Center and all devices on the Smart Energy network must support the installation and use of these keys as described in the security section.

Other devices within a home may also be capable of receiving public pricing information and messages from the metering network. These devices may not have or need all the capabilities required to join a Smart Energy network. Mechanisms are provided to publish public pricing data and messages to these devices without requiring they join the Smart Energy network. These mechanisms are described in the sections describing both the public pricing and message exchanges.

Metering networks are primarily installed by specialized service personnel, but other devices in the network may be added by home owners, or home automation professionals who may not have any ZigBee expertise. Installation concepts must be easy and uniform across Smart Energy device manufacturers.

Smart Energy networks could include both ZigBee 2007 and ZigBee 2007 Pro nodes. It is recommended the majority of the nodes in the network should be

based on one stack profile or the other to get consistent performance. ZigBee Smart Energy certified products must be based upon a ZigBee Compliant Platform (ZCP). If the Smart Energy profile resides in conjunction with a private profile, the product should be ZigBee Manufacturer Specific Profile (MSP) certified and must be Smart Energy ZCP certified. This additional certification provides a reassurance that the underlying stack is behaving properly and the application is not abusive to the network.

Smart Energy networks will not interact with a consumer ZigBee Home Area Network unless a device is used to perform an “application level bridge” between the two profiles or the HA devices satisfy the Smart Energy profile security requirements. This is due to the higher security requirements on the Smart Energy network that are not required on a Home network. However, it is expected that Home Automation devices that are extended to include the Smart Energy profile can still operate in a home network.

The ZigBee Smart Energy Network makes possible networks such as the following:

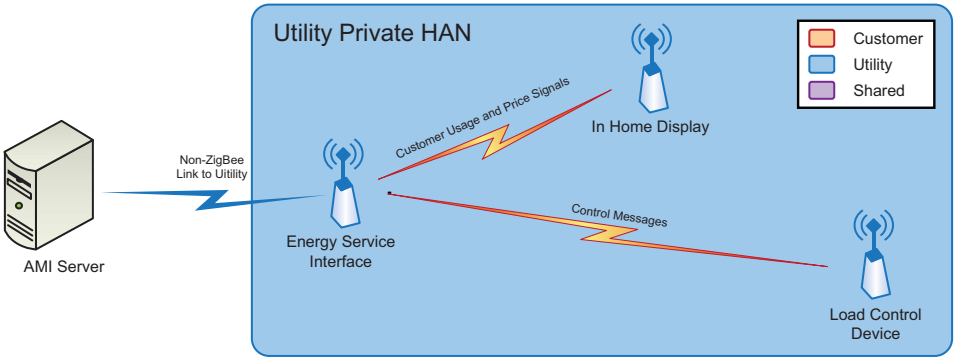

Figure 5.1 Utility Private HAN

Utility Private HAN might include an in-home display, or a load control device working in conjunction with energy service interface[4], but it would not include any customer controlled devices.

4. CCB 1072

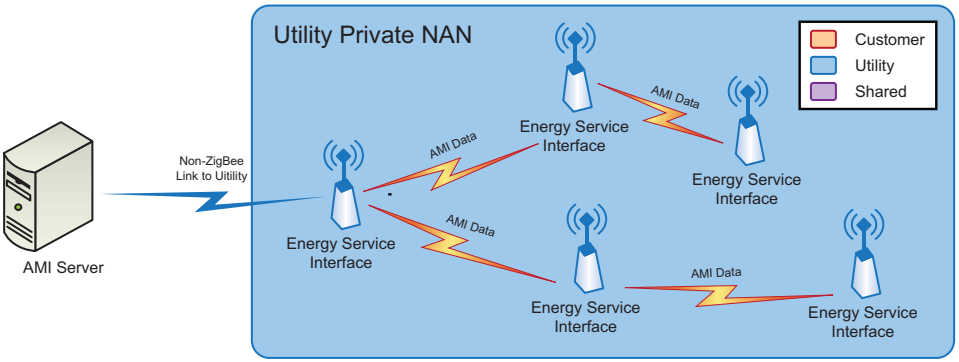

Figure 5.2 Utility Private NAN

Utility Private ZigBee network might also be used as a NAN, where ZigBee provided the primary communications for a Smart Energy deployment.

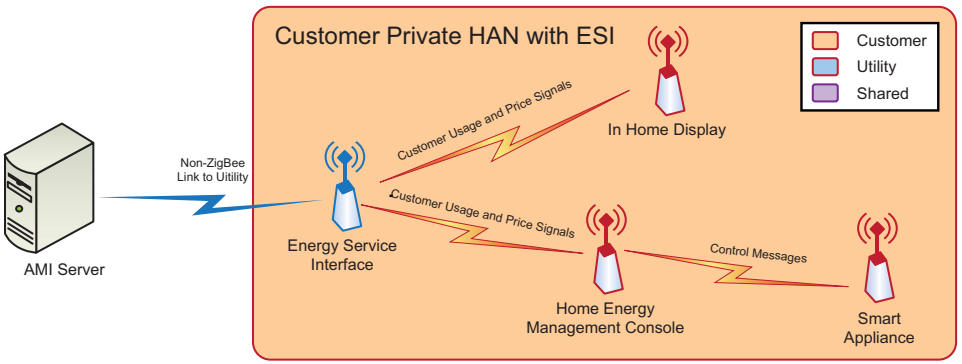

Figure 5.3 Customer Private HAN

ESI[5] provided by utility, but limited to the role of information provider (Usage and Pricing) into a customer HAN that utilizes an Energy Management Console for conveying or controlling local devices. An example is controlling a smart appliance based upon a pricing signal.

5.2 ZigBee Stack Profile

Products that conform to this specification shall use stack profile number 0x01 or profile 0x02, as defined in [B7]. In addition to the requirements specified in [B7], the following requirements are mandatory for this application profile.

5. CCB 1072

- Support for Application link keys is required.
- Fragmentation is required. Please refer to 5.3.8 regarding fragmentation sizes and parameter settings.[6]

5.2.1 MAC Data Polling (NMLE_Requests)[7]

MAC Data polling is required by all sleepy end devices to operate correctly in a ZigBee Pro network. Smart Energy puts no restrictions on the frequency of MAC data polls. The choice of how frequently data polling is done will be based on individual product design considerations to reduce power consumption. However the following are a set of recommendations to insure correct operation in the network:

- The MAC data polling rate should be dynamic based on the device's operating state. It is recommended it has at least two rates, a fast rate and a slow rate.
- The ZigBee specification only requires that parent devices buffer a single message for 7.5 seconds. This single buffer applies to all sleepy end devices. Therefore a sleepy device should poll more frequently than once per 7.5 seconds in order to be able to retrieve a buffered message.
- When the device is waiting for an active response message such as an APS acknowledgement, or a ZCL response, or participating in a multi-message protocol, it should poll at its fast rate. This fast rate is recommended to be at least once every 3 seconds.
- When the device is not actively waiting for messages it can poll at its slow rate. For example once per hour. This insures it still has a connection with the network and with its parent.

During initial joining to the Smart Energy network, including key establishment and service discovery, it should poll at its fast rate.

5.2.2 Application Level Queries[8]

It is expected that client devices will periodically send application level queries to servers to retrieve data. This may be done for example by thermostats querying the current price, or an in-home[9] display to show the current reading of a meter.

Due to the fact that all ZigBee devices within the HAN utilize a shared medium for sending and receiving data it is recommended that devices do not saturate the

6. CCB 1267 - Deleted two bullets
7. CCB 1267
8. CCB 1267
9. CCB 1570

network with frequent queries for data that does not change often. As a general rule, but not a requirement, it is recommended that devices do not initiate more than 1 query per second. This recommendation does not apply to responses generated locally due to the receipt of remote device requests. In addition, it is possible that the device may need to generate a burst of traffic and exceed this recommendation. This bursting period should be very limited and followed by a period of reduced traffic respecting the above guidelines.

5.2.3 ZigBee Coordinator and Trust Center Recommendations

- In a Smart Energy based HAN network the Trust Center shall be the Coordinator (short address 0x0000).
- In a Smart Energy based HAN network the Trust Center shall be an ESI in the network.[10]
- In a Smart Energy based NAN the backhaul point is likely to be the coordinator and trust center.

5.3 Startup Attribute Set (SAS)

In order to insure interoperability, all ZigBee Smart Energy devices shall implement compatible Startup Attribute Sets (SAS) as defined in this specification. This does not mean that the set must be modifiable through a commissioning cluster, but that the device must internally implement these stack settings to insure compatibility and consistent user experience. The startup set parameters described by the commissioning cluster in [B2] provide a good basis to specify a Smart Energy start up set.

Because Smart Energy Devices are likely to be preconfigured at a warehouse and installed by a technician, a specific start up set values may be established by a particular utility or service area and these startup set values used in place of these below for installation. The startup set values that would be expected to be set by the installer are noted below.

5.3.1 Startup Parameters

The startup parameters and their default values are listed in Table 5.1.

10. CCB 1072

Table 5.1 Startup Parameters

Parameter	Value	Comment
Short Address	0xFFFF or installer specified.	
E PANiD	0x0000000000000000 or installer specified.	
PAN ID	0xFFFF or installer specified.	
Channel Mask	All channels in frequency band.	If needed, the power transmitted by the device on channel 26 can be lowered to comply with FCC regulations.
Protocol Version	0x02 (ZigBee and later)	
Stack Profile	1 (ZigBee) or 2 (ZigBee PRO)	
Startup Control	2 (two) if un-commissioned, so it will join network by association when a join command is indicated. 0 (zero) if commissioned. Indicates that the device should consider itself a part of the network indicated by the *ExtendedPANId* attribute. In this case it will not perform any explicit join or rejoin operation.	
Trust Center Address	0x0000 (short id) installer specified Eui64.	Please note: In Smart Energy Profile 1.1 and above, only the Coordinator (0x0000) can be the SE Trust Center.[a]
Master Key		Not used, high security is not used in this profile.
Link Key	0x00000000000000000000000000000001 if the Key Establishment Cluster is being used to install a link key Installer provided if using preconfigured link keys	
Network Key	0x00000000000000000000000000000001 if no pre-installed key present	
Use Insecure Join	0x00 (False)	Flag that disables the use of insecure join as a fallback case at startup time

a. CCB 1284

5.3.2 Join Parameters

The join parameters and their default values are listed in Table 5.2.

Table 5.2 Join Parameters

Parameter	Value	Comment
ScanAttempts		At boot time or when instructed to join a network, the device should complete up to three (3) scan attempts to find a ZigBee Coordinator or Router with which to associate. If it has not been commissioned, this means that when the user presses a button or uses another methodology to join a network, it will scan all of the channels up to three times to find a network that allows joining. If it has already been commissioned, it should scan up to three times to find its original PAN to join. (ZigBee Pro devices should scan for their original extended PAN ID and ZigBee (2007) devices can only scan for their original PAN ID).
TimeBetween Scans	1 second	Determines the number of seconds between each scan attempt.
RejoinInterval	60 seconds or shorter	How quickly a device will attempt to rejoin the network if it finds itself disconnected.
MaxRejoinInterval	15 minutes	Imposes an upper bound on the RejoinInterval parameter - this must be restarted if device is touched by human user, i.e. by a button press. This parameter is intended to throttle how often a device will scan to find its network in case the network is no longer present and therefore a scan attempt by the device would always fail (i.e., if a device finds it has lost network connectivity, it will try to rejoin the network, scanning all channels if necessary). If the scan fails to find the network, or fails to successfully rejoin, the device will wait for 15 minutes before attempting to rejoin again. To be network friendly, it would be recommended to adaptively extend this time period if successive rejoins fail. It would also be recommended the device should try a rejoin when triggered (via a control, button, etc.) and fall back to this interval if rejoins fail again.

5.3.3 Security Parameters

The security parameters and their default values are listed in Table 5.3.

Table 5.3 Security Parameters

Parameter	Value	Comment
SecurityTimeoutPeriod	Set by stack profile.	
TrustCenterNetworkKey	The Trust Center will pick the network key.	ZigBee Smart Energy devices shall depend on either pre-configured keys to be commissioned or the use of the Key Establishment Cluster with a pre-configured Trust Center link key to get the network key (not in the clear). ZigBee Smart Energy networks will not generally send keys in the clear.

5.3.4 End Device Parameters

The end device parameters and their default values are listed in Table 5.4.

Table 5.4 End Device Parameters

Parameter	Value	Comment
IndirectPollRate	Set by stack profile	This is how often a device will poll its parent for new data. It is recommended that an end device that is designed to receive data should poll its parent every 60 seconds.

5.3.5 Link Status Parameters

The link status parameters and their default values are listed in Table 5.5.

Table 5.5 Link Status Parameters

Parameter	Value	Comment
LinkStatusPeriod	Set by stack profile	
RouterAgeLimit	Set by stack profile	
RepairThreshold	Set by stack profile	

5.3.6 Concentrator Parameters

The concentrator parameters and their default values are listed in Table 5.6.

Table 5.6 Concentrator Parameters

Parameter	Value	Comment
ConcentratorFlag	Set by stack profile	Identifies the device to be a concentrator.
ConcentratorRadius	11 (eleven)	Device manufacturers that produce a concentrator product will set the max concentrator radius to this value.
ConcentratorDiscoveryTime	Set by stack profile	Identifies how often the Concentrator network layer should issue a route request command frame.

5.3.7 APS Transport Parameters

The APS transport parameters and their default values are listed in Table 5.7.

Table 5.7 APS Transport Parameters

Parameter	Value	Comment
MaxFrameRetries	Set by stack profile	This determines the maximum number of retries allowed after a transmission failure.
AckWaitDuration	Set by stack profile	This is the maximum number of seconds to wait for acknowledgement of an APS frame.

5.3.8 APS Fragmentation Parameters

For fragmentation there are application settings from the APS IB that must be defined by the application profile. For Smart Energy these parameters are to be set as shown in Table 5.8.

Table 5.8 APS Fragmentation Parameters

Parameters	Identifier	Type	Value	Description
apsInterframe Delay	0xc9	Integer	50	Standard delay in milliseconds between sending two blocks of a fragmented transmission (see [B3] sub-clause 2.2.8.4.5)
apsMaxWindowSize	0xcd	Integer	1	Fragmentation parameter – the maximum number of unacknowledged frames that can be active at once (see [B3] sub-clause 2.2.8.4.5).

In addition the Maximum Incoming Transfer Size Field in the Node descriptor defines the largest ASDU that can be transferred using fragmentation. For the Smart Energy Profile the default value shall be set to 128 bytes. Maximum ASDU size allowed is specified in [B3] and dictated by solution needs and RAM capacities of the communicating devices.

It is highly recommended all devices first query the Node Descriptor of the device it will communicate with to determine the Maximum incoming transfer size (if ASDU size is greater than 128 bytes). This will establish the largest ASDU that can be supported with fragmentation. The sending device must use a message size during fragmentation that is smaller than this value.

5.3.9 Binding Parameters

The binding parameters and their default values are listed in Table 5.9.

Table 5.9 Binding Parameters

Parameter	Value	Comment
EndDeviceBindTimeout	60 seconds	Timeout value for end device binding. End Device binding is set by the coordinator.

5.4 Smart Energy Profile Security

To be part of a Smart Energy network, a device shall associate using one of the two association methods described below and require the use of the Key Establishment Cluster (see Annex C) for installation and updating of link keys.

All devices shall have the ability to retain their joining and security settings through power outages.

5.4.1 Joining with Preinstalled Trust Center Link Keys

When using preinstalled trust center link keys, the following steps are used:

1. Trust Center link keys SHALL be[11] installed in each device prior to joining the utility network.
2. The trust center link key for a device that is to be joined SHALL be[12] provided to the local trust center through an out of band means as described in sub-clause 5.4.8.1 “Out of Band Pre-Configured Link Key Process”.

11. CCB 1403
12. CCB 1403

3 Permit joining is turned on in the network. The Trust Center enables joining by calling the NLMEPERMIT-JOINING.request primitive. Joining must be managed for an appropriate amount of time but SHALL NOT be broadcast with a time of greater than 254 seconds should not repeatedly broadcast with out hearing device announcement or network administrator action.[13] The appropriate amount of time will be dictated by the overall performance of the system and business processes driving the registration and device authorization activities. See sub-clause 5.4.1.2, “Best Practice for Coordinator Permit Joining Broadcasts”.

4 Be aware Joining has an internal time out within the ZigBee stack, therefore joining may need to be enabled multiple times during the overall Registration and device authorization process.[14]

5 A device autonomously joining a network (i.e. without user supervision or input) may initially scan for networks to join three times in succession without pausing. After failing to successfully join a network, the device SHALL exponentially increase time between scan times, eventually performing a channel scan at a maximum rate of once per hour. The device may increase scan rate upon request from user input, such as a button push or power cycle.

6 The device joins the network and is sent the network key encrypted with the key-transport key derived for the preinstalled trust center link key. The procedure for doing this is detailed in Annex F, also reference [B3] section 4.5.4 on key-transport keys and [B3] section 4.4.1 on frame security for the APS layer.

7 The trust center must update the pre-configured trust center link key in the joining device using the Key Establishment Cluster after completion of the joining procedure.

8 The trust center of the network has the option of later updating the trust center link keys with devices in the network as desired by the application using the Key Establishment Cluster. Updating security keys should be an infrequent operation.

9 Once joining is completed, the list of authorized devices in the Trust Center should be updated, please refer to sub-clause 5.4.1.1, “Best Practices for Tracking Registered Devices”.[15]

5.4.1.1 Best Practices for Tracking Registered Devices

In order to properly track Smart Energy Devices and communicate device registration status to upstream systems, Trust Centers (ESIs)[16] should maintain a

13. CCB 1403
14. CCB 1403
15. CCB 1403

list of authorized devices. It is also recommended that Trust Centers maintain the following items for each of the registered devices:

1. Client EUI64
2. Client Installation Code
3. Registration Status
4. Time and Date Stamps

Although this information is not exposed through the ZigBee network, device binding is expected to be used to track and understand ZigBee network connectivity.[17]

5.4.1.2 Best Practice for Coordinator Permit Joining Broadcasts

It will be left to the coordinator / administrators of the network to determine when a network should be allowing joining. However when the network is allowing joining:

1. At the start of the joining period the coordinator will allow joining and broadcast a permit join message for the lesser of the permit join period or 254 seconds.
2. Every 240 seconds or whenever a device announce is received the coordinator will broadcast a permit join message for the lesser of the remaining permit join period or 254 seconds. Administrators of a network shall try to keep the amount of time devices on their networks allow joining to a minimum.

Note: *sending out a permit join message with a time of 255 (forever) is disallowed due to the risk of not being able to reliably tell devices to stop permitting joining in the future.*

5.4.2 Re-Joining a Secured Network

5.4.2.1 Rejoining Node Operation

When a device is re-joining a secured network, the following steps are used:

1. Permit joining is not required to be on in the network.
2. The device shall attempt a rejoin using the procedure detailed in [B3] Section 3.6.1.4.2 with network security. The network key and sequence number used will be the ones previously obtained from the trust center.

16. CCB 1072
17. CCB 1159

3 If the secured rejoin is successful, nothing more is required from the device.

4 If the secured rejoin fails, the device shall attempt a rejoin using the procedure detailed in [B3] Section 3.6.1.4.2 without network security. The re-joining device is assumed to have previously joined the network and obtained a link key using the key establishment cluster procedures. If the device does not have a link key obtained via the key establishment cluster, it cannot rejoin the network.

5 If the [18]rejoin fails the device may attempt it again. If the device is told to leave the network it may employ the Joining using the Key Establishment Cluster procedure.

5.4.2.2 Trust Center Operation

When the trust center receives notification that a device has rejoined the network, the following steps are used:

1 If the device performed a secured rejoin the trust center is not required to take any action.

2 If the device performed a[19] rejoin[20] the trust center shall determine if the device is authorized to be on the network. [21]The trust center should send out an updated copy of the network key encrypted with the corresponding link key.

3 If the trust center determines that the device is not authorized to be on the network, it shall send an APS *Remove Device* command to the parent of the rejoining device, with the target address of the rejoining device's IEEE address. The parent will then remove that device from its child table.

Note: *The Trust Center and Router behaviors described in sections beginning at sub-clause 5.4.2.2.1 up until sub-clause 5.4.3 in this revision of this specification are provisionary and not certifiable. This text may change before reaching certifiable status in a future revision of this specification.*

5.4.2.2.1 Initiating Re-Registration[22]

Note: *The Trust Center and Router behaviors described in sections beginning at sub-clause 5.4.2.2.1 up until sub-clause 5.4.3 in this revision of this specification are provisionary and not certifiable. This text may change before reaching certifiable status in a future revision of this specification.*

18. CCB 1026
19. CCB 1026
20. CCB 1592
21. CCB 1482
22. CCB 1159

To initiate the re-registration process for a device, the Trust Center (ESI) would invalidate the Link keys for that device and subsequently cause a re-authentication / authorization to re-establish Link Keys. The processes required for this activity are:

1. The Trust Center invalidates the Link key by using the APSME-SET primitive.
2. When the Client device detects communication errors due via APS error results or by experiencing multiple re-try failures, both caused by the invalid Link Keys, it starts the processes to validate the following conditions:
   a. The Device validates its still part of the network.
   b. Route discovery processes validate communications paths are still in place.
3. If both conditions are true, the Client device attempts a secure re-join outlined in Re-joining a Secured Network and subsequently refreshes the Link Keys.
4. Re-binding of services take place (if needed).
5. Once Registration is completed, the list of authorized devices in the Trust Center should be updated, please refer to sub-clause 5.4.1.1.[23]

5.4.2.2.2 Initiating De-Registration

To initiate the de-registration process for a device, which is the process of removing a previously registered device, the Trust Center (ESI) would use the following processes for this activity:

1. The Trust Center (ESI) invalidates the Link key by using the APSME-SET primitive.
2. The Trust Center (ESI) informs the Client device to leave the network by calling the NLME-LEAVE.request primitive.
3. The Trust Center (ESI) informs any Routers to remove the Client device by calling the APSME-REMOVEDEVICE.request
4. The ESI would unbind any services associated with the Client device by calling the APSME-UNBIND primitive.
5. Once de-registration is completed, the list of authorized devices.[24]

5.4.2.2.3 Trust Center Swap-Out

Note: *The Trust Center Swap-Out feature in this revision of this specification is provisionary and is not certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.*

23. CCB 1159
24. CCB 1159

This section describes the requirements for swapping out a Trust Center in a Smart Energy 1.0 network. In SE 1.0, an ESI should act as the coordinator and trust center of the network. In most deployments the ESI is the meter and therefore the TC. There can only be one TC in a SE network, although multiple ESIs may exist on the network. The TC (ESI) in a SE network is responsible for performing authentication and authorization. SE devices which are allowed to join the network are provisioned on the TC (ESI) from the head-end over the utility's backhaul connection.

When a TC is replaced the new device is given the extended PAN ID of the previous network, and the addresses and associated trust center link keys of all the devices from the previous network. Both the existing devices and the TC treat these keys like installation codes (unauthorized), which have limited privileges in the network. Once the devices successfully connect to the new trust center they must re-establish new TC link keys using CBKE.

The existing SE 1.0 devices must be upgraded to include behavior that allows them to detect a failure to communicate with the existing trust center. When it detects this condition a device will go off in search of another network with the same extended PAN ID as the current one. If a network is found then the device will perform a first time join using the [25]NWK rejoin and its current TC link key as the pre-configured key. If the device is able to successfully join the network then it will immediately initiate CBKE to derive a new link key with the replacement TC. If that succeeds the device will identify the device's IEEE as the identity of the new trust center, and begin operating in the new network; the device shall locate any services that it may have been using.[26]

If it is unable to join to the new network or unable to successfully negotiate CBKE, then the device will return to its previous network and continue operating. If the trust center is still unreachable at a later point in time it can perform the above steps again to attempt to find a new network.

Trust Center Swap-Out is an optional feature and is not required for ZigBee Smart Energy Certification.[27]

5.4.2.2.3.1 SE Router Requirements

All routers in the network shall be able to identify when the trust center is no longer accessible in the network. This will be done by periodically sending an APS datagram to the Trust Center and receiving the APS acknowledgment. The APS datagram shall require encryption and acknowledgement.

25. CCB 1026
26. CCB 1419
27. CCB 1241

After an extended period where multiple attempts have been made to contact the trust center and failed to get a response, a device would temporarily drop off the network to go in search of a network where the trust center was present. The new network may have different network parameters than the old one, but the extended PAN ID value would always be the same. Those networks that match all of the parameters of the old network will be filtered out in preference of a new network with one or more different parameters. This enables the device to find a newer instance of the existing network.

Once a new instance of the existing network has been found, the device would perform various procedures to attempt to join that network and authenticate with the new trust center device. If at any point during the attempt a failure occurred, the device may continue scanning for networks to join or return to its existing network and continue operating as it had before.

As a last resort, all devices must have a means to return to factory defaults so that they can be recomissioned. This would involve reverting back to the use of an installation code and forgetting all previous network and application parameters. This provides a means to reconnect the device to the existing network when other methods have failed, or decommission the device and join it to a new network.

5.4.2.2.3.2 Per SE Network Storage Requirements

TC swap-out requires the backup of data to an off-chip device. The data and storage requirements are listed in Table 5.10. Backup of the Extended PAN ID should be performed once the ESI has been commissioned or the network is formed. Backup of the TC Link Key Hash (see sub-clause 5.4.2.2.3.6) should be performed on successful completion of CBKE with the TC. TC Link Key updates from subsequent CBKE shall also be backed up. The Install Code derived TC Link Key may be backed up when the device is provisioned on to the SE network.

Table 5.10 Per SE Network Storage Requirements

Data Description	Number of Bytes	Mandatory / Optional
Extended PAN ID	8 bytes[a]	M
Registered device EUI64	NumberOfDevices * 8 bytes	M
Registered device Hashed TC Link Key	NumberOfDevices * 16 bytes	M
Registered device Install Code	NumberOfDevices * 16 bytes	O

a. CCB 1275

5.4.2.2.3.3 Utility Requirements

It is expected that the utility is able to store backup data about each Smart Energy network in order to facilitate the TC swap-out feature. It is recommended that the

list of IEEE (EUI64) addresses of devices registered in the network, and their associated installation code, always be backed up. This will help to deal with an unexpected situation due to the customer or the utility, which requires one or more HAN devices to be recommissioned.

5.4.2.2.3.4 Keep Alive Method

In order to detect the TC is no longer available all SE routers shall implement a keep-alive mechanism with the TC. The Key Establishment cluster is mandatory on all SE devices. The SE routers shall send an APS encrypted ZCL message on a periodic interval of up to a maximum of 20 minutes. The minimum polling rate should not be less than 5 minutes. Failure to receive an encrypted APS data frame (such as a read attribute response) shall indicate the TC is no longer available. If the device fails to receive 3 APS encrypted data frames in a row it shall consider the TC no longer accessible and initiate a search for it.[28] Failure of the encryption or frame counter shall constitute a failure of the keep-alive.

5.4.2.2.3.5 Trust Center Swap-out Process

The following steps describe the Trust Center swap-out process.

Preconditions: ESI installed and PAN formed.

1 Back up Extended PAN ID to off-chip device (mandatory).

2 SE device provisioned on the ESI and installed.

3 Back up EUI64 and Install Code to off-chip device (optional).

4 SE device performs CBKE successfully, derived TC link key and EUI64[29] backed up to off-chip device (mandatory).

5 Any updates to the TC Link Key must be backed up to off-chip device (mandatory).

6 SE device sends periodic APS encrypted command to the TC.

7 ESI replaced with the Extended PAN ID, list of EUI64s and hashed TC link keys restored from backed up data. (Permit joining is not required to be on in the network). The TC link keys shall be treated as install code-derived link keys and unauthorized.

8 New TC forms a new network using new network key, new short PAN ID, and backup of extended PAN ID.

9 SE device detects TC no longer available (see sub-clause 5.4.2.2.3.4).

28. CCB 1491
29. CCB 1285

a. There are 4 possible cases at this point.
   i. The TC is temporarily unavailable.[30]
   ii. The device missed a network key update.[31]
   iii. The TC changed channels to avoid congestion.[32]
   iv. The TC has been swapped out.[33]

10. SE device scans for the current Extended PAN ID (the short PAN ID will probably have changed).[34]

11. Prior to performing a [35]rejoin the device must backup in local storage its current TC link key, state of the link key (authorized or unauthorized), network key, and associated NWK and APS frame counters.

12. If the TC sends a Transport Key message encrypted using the device's existing TC link key and the device is able to successfully[36] decrypt and rejoin the network, no further operations are necessary. The device can resume all normal operations.

13. If the TC sends a Transport Key message encrypted using a 128-bit AES-MMO hash of the TC link key, then the device shall ignore the frame counter check and accept the new network key. It will also record the source IEEE address of the sending device as the new TC address. It shall mark the hashed TC link key as not authorized and treat the link key as an installation code. It must now perform Key Establishment to fully authenticate itself in the network.

14. If the key establishment is NOT successful the device may try again immediately. Otherwise it shall leave that network. It can continue scanning for additional networks to rejoin, or restore the values of its previous network and resume normal operation.

15. If the Key Establishment is successful then the device can resume normal operation. It may discard the backup of security data from the old network. The TC must backup hash of the new TC Link Key for this device to an off-chip device.

16. If after attempting rejoin with all discovered PANs fails then the device shall fall back to the existing PAN.

30. CCB 1285
31. CCB 1285
32. CCB 1285
33. CCB 1285
34. CCB 1286
35. CCB 1026
36. CCB 1294

5.4.2.2.3.6 Link Key Hash

In order to protect the data that is being backed up, a hash on the TC link key will be performed and that will be the key stored externally. It is highly recommended that the actual link key used for operational networks never be transported out of the ESI. Using this method if the backup data for the TC is compromised then it cannot be used to compromise existing ZigBee network communications.

The hashed key shall be created by performing a 128-bit AES-MMO hash on the 128-bit key data. The following is a test vector for the hash:

Table 5.11 Example Hash of the TC Link Key

TC Link Key	C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF
Hashed TC Link Key	A7977E88BC0B61E8210827109A228F2D

5.4.2.2.3.7 Trust Center (ESI/Meter)

Dependencies

The ESI shall support backup and restore of data (including TC link keys) to an off-chip device.

Routers shall detect the TC is no longer available by sending an APS encrypted command and receiving the APS acknowledgement with a maximum periodic interval of 20 minutes.

Table 5.12 Parameters of Trust Center Swap-Out

Name	Type	Range	Default	Mandatory / Optional
TC Keep-Alive	Unsigned 8-bit integer	0x01 - 0x1E	0x14	M

5.4.3 Devices Leaving the Network

Upon receipt of an APS update device command indicating a device has left the network the trust center shall not remove the trust center link key assigned to that device. This is to prevent a device on the network performing a denial of service attack by spoofing the MAC address of another node and issuing a false ZigBee *Network Leave* command. Devices should be removed from Trust Center authorization and trust center link key lists via out of band methods, i.e. secure meter back haul or secure IP interface.[37]

37. CCB 1404

Devices should follow the guidelines for stale keys described in 5.4.5.[38]

5.4.4 Updating the Network Key

Periodically the trust center shall update the network key. This allows the trust center to phase out a previous instance of the network key so that devices that are no longer on the network will not be able to perform a secure rejoin. Those devices must then perform a [39]rejoin, which allows the trust center to authorize whether or not they are allowed to be on the network.

When the trust center wishes to update the network key it will broadcast the network key to all devices in the network. All devices receiving the key update will store but will not start using the new key.

It is assumed that routers will receive the network key update sent by the Trust Center. Sleepy end devices are unlikely to get the network key update sent by the Trust Center unless the device polls frequently.

After sending an updated network key, the trust center shall wait a minimum of nwkNetworkBroadcastDeliveryTime before sending the switch key message. Devices that miss the key switch broadcast message will implicitly switch when they receive any network message that is encrypted using the new key sequence number.

Once the network has started using the new key, any device that has missed the key update message will not be able to communicate on the network. Those devices that missed the key update must follow the Re-joining a Secured Network procedure.[40]

5.4.5 Updating the Link Key

Periodically the trust center may update the link key associated with a particular device. This allows the trust center to phase out the existing key and refresh it with a new key. The trust center can decide on its own what the policy is for how long a link key may be used and how often it should be updated.

Trust Center link keys are used for sending application messages as well as stack commands. Therefore a trust center cannot simply delete a link key that it wants to update. The trust center must accept and or send encrypted APS commands to or from a device even if has retired that link key from encryption of application data messages. This is especially necessary for sleeping end devices, which may not

38. CCB 1159
39. CCB 1026
40. CCB 1059

have the current network key and need to use their link key to obtain an updated copy during a rejoin.

When the trust center deems that a particular link key should no longer be used, it shall mark the key as stale. A stale key shall not be used to send data messages. Devices that receive a message using a stale key should discard the message and shall not send an APS acknowledgement to the sender.

Devices shall accept and process APS commands that are encrypted with a stale key.

When the trust center receives a message encrypted with a stale link key, it shall initiate the key establishment procedure to negotiate a new link key. Upon successful establishment of the new link key with the device, the device shall clear the stale indicator for that key.

Devices that are not acting as the trust center may utilize their own policy for retiring and updating application link keys with other devices that are not the trust center. Those devices are not required to keep around retired keys and therefore may delete them prior to establishing a updated link key using the key establishment cluster.

5.4.5.1 Network Joining and Registration Diagram

Figure 5.4 depicts an example of a successful network startup and certificate exchange (with pre-established link keys). Please refer to Annex C for further discussions on communication exchanges and key support.

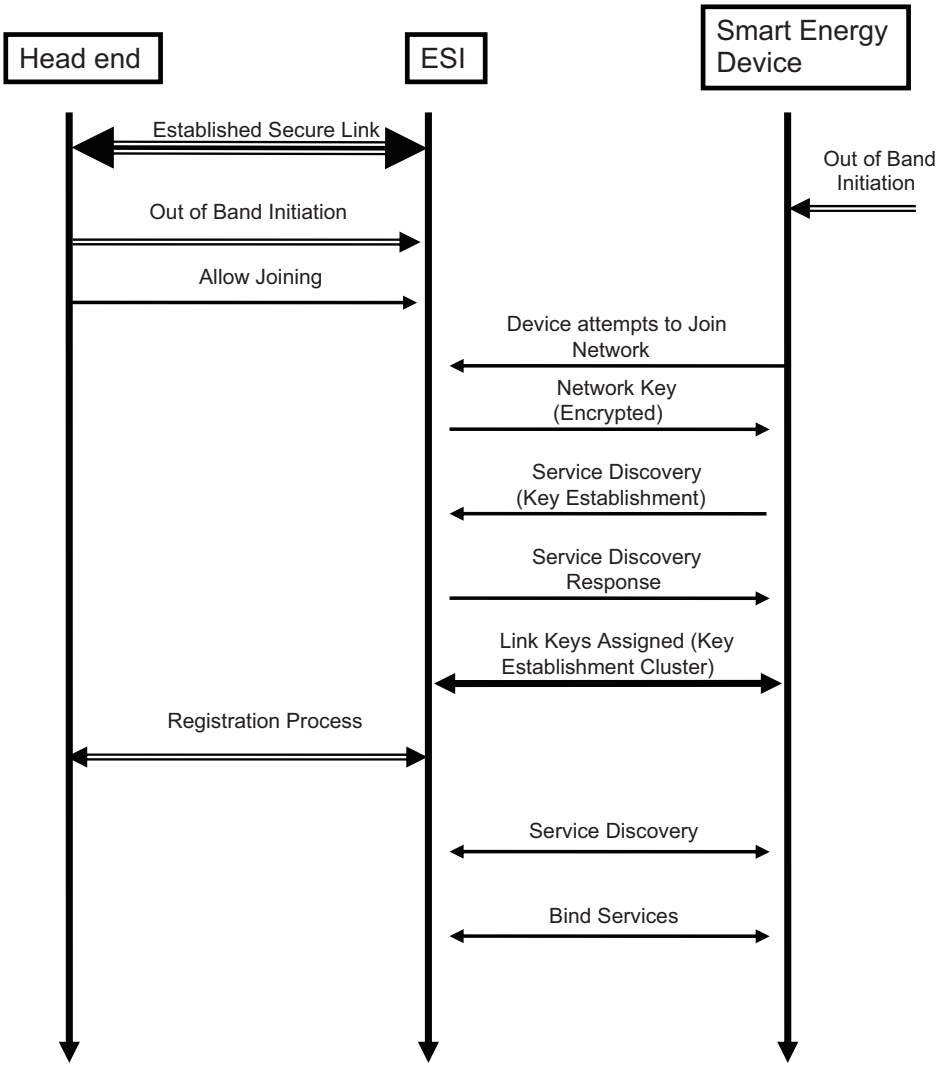

Figure 5.4 Successful Join and Registration

Please note: After joining the network and acquiring a Network Key, the Smart Energy End Device shall initiate the Service Discovery process to locate the Key Establishment Cluster. As recommended best practice, the ESI[41] should support a fault-tolerant behavior by initiating Key Establishment Cluster service discovery process whenever it detects the Smart Energy End Device fails to do[42] so.

41. CCB 1072
42. CCB 965

After Joining and after Key Establishment:[43]

- Client SHALL perform service discovery.
- Sleepy devices SHALL perform “get” requests for data they wish to receive and SHOULD NOT expect to receive unsolicited messages.
- If a Client wishes to receive unsolicited messages, Client SHALL follow with attempt(s) to ZDO Bind Request. A Client does not have to support a binding table.
- If Server does not support binding, Server SHALL perform service discovery and register those devices for unsolicited messages (whether or not they want the messages).
- For backward compatibility, Server SHOULD perform service discovery and register those devices for unsolicited messages (whether or not they want the messages).

5.4.6 Cluster Usage of Security Keys

The SE Profile utilizes a higher level of security on the network but not all clusters need to utilize Application Link keys. All clusters are required to use network layer encryption using the network key.[44] Table 5.13 identifies the security keys utilized by each cluster:

Table 5.13 Security Key Assignments per Cluster

Functional Domain	Cluster Name	Link Key Required[a]
General	Basic	No
General	Identify	No
General	Alarms	No
General	Time	Yes
General	Commissioning	Yes
General	Power Configuration	No
General	Key Establishment	No
Smart Energy	Price	Yes
Smart Energy	Demand Response and Load Control	Yes
Smart Energy	Metering[b]	Yes

43. CCB 1130
44. CCB 1397

Table 5.13 Security Key Assignments per Cluster (Continued)

General	Over the air Bootload Cluster	Yes
Smart Energy	Messaging	Yes
Smart Energy	Tunneling and Generic Tunneling	Yes
Smart Energy	Prepayment	Yes

a. CCB 1397

b. CCB 940

Once a Registered SE device has an Application Link Key established with the ESI[45], it may also establish Application Link Keys with any other device on the SE Network. This is accomplished by using the ZigBee service and device discovery process (employing the Network Key). Regardless of the communication paths, all SE applications shall use and validate the Security key usage as listed in listed in Table 5.13. If link key encryption is NOT used but required, the receiving device shall generate a ZCL Default Response, employing the Network Key, with a FAILURE (0x01) status code.[46]

It is permissible for a device to initiate a ZCL exchange using an application link key even when not required. If a device receives a message with link key security even though it is not required as per Table 5.13, it shall accept the message. Additionally, if a response is sent then it shall use link key encryption.[47]

5.4.7 Key Establishment Related Security Policies

The following are the policies relating to Key Establishment that are recommended for Smart Energy networks.

5.4.7.1 Joining

If the device doesn't need to perform discovery queries or other non-secure operations after it joins an SE network and receives the Network Key, it should immediately initiate Key Establishment with the Trust Center to obtain a new Trust Center Link Key.

If Key Establishment fails with a result of UNKNOWN_ISSUER the device should leave the network. A device that does not initiate Key Establishment with the Trust Center within a reasonable period of time may be told to leave depending on the network operator’s policy.[48]

45. CCB 1072
46. CCB 1397
47. CCB 1397

Upon successful negotiation of a new Trust Center Link Key the device may communicate using clusters that require APS security.

5.4.7.2 Trust Center

The Trust Center shall keep track of whether a particular device has negotiated a CBKE Trust Center Link Key, or whether only a preconfigured Trust Center Link Key exists. The Trust Center shall not use the preconfigured link key to send encrypted APS Data messages to the device. The Trust Center shall discard any APS encrypted APS Data messages that use the preconfigured link key, and it shall not send APS Acks for those messages.

The Trust Center shall accept and send APS Data messages that do not use APS Encryption to a device that has not negotiated a CBKE Trust Center Link key provided that the security usage for that cluster allows using only Network layer security (encrypted with the Network Key). See sub-clause 5.4.6, “Cluster Usage of Security Keys”.

The Trust Center is required to be a Smart Energy device. It is required to support key establishment server on at least one endpoint, though it may support it on more than one endpoint. These endpoints shall be considered to all refer to the same logical ZigBee device type, in other words the Trust Center. Any negotiation or establishment of a link key on one endpoint applies globally to the Trust Center as a device and is not specific to an endpoint.

The Trust Center shall have a means of adding and removing keys of specific devices that are part of the Smart Energy network. The specific means of doing this is outside the scope of this document.

5.4.7.3 During Joining

Normal operation of a device in a Smart Energy network requires use of a preconfigured link key, established by using the Installation Code (refer to sub-clause 5.4.6), to join a ZigBee Pro network. After joining the network a device is required to initiate key establishment using ECMQV key agreement with the Trust Center, to obtain a new link key authorized for use in application messages.

Prior to updating the preconfigured link key using key establishment, the Trust Center shall not allow Smart Energy messages that require APS encryption. Although the node has a link key, that node has not been authenticated and thus the key's use is not authorized for application messages. Its use is still required for certain stack messages (e.g., the APS Command Update Device) and must be accepted by the trust center.

In order to perform key establishment the device must discover an endpoint on the Trust Center that supports the key establishment server cluster. The joining device

48. CCB 1286

shall perform a ZDO Match Descriptor Request to determine what endpoint to use. This request shall be unicast to the Trust Center's short address of 0x0000. When a reply is received it may contain multiple endpoints that indicate support for the key establishment server. The joining device may use any endpoint to perform key establishment. Link keys established using key establishment are global to the Trust Center device and are not specific to a particular endpoint.

Once a node has been authenticated by the Trust Center and obtained an authorized link key using key establishment, it may communicate with the Trust Center using APS layer security. The Trust Center should accept valid APS encrypted message using that new link key. At this point the joining device can communicate to the Trust Center as a Smart Energy device.

If a device never establishes a trust center link key after joining, the trust center may send it a network leave command. This is only done for non-security reasons, such as encouraging a well-behaved device that it is not on the correct network. Malicious nodes may be forced off the network by having the Trust Center send a unicast update of network key followed by a broadcast switch key.[49] However, if the above mentioned security policies are adhered to, then the malicious node will be unable to communicate at the application level[50] with other devices since it will not have access to an authorized link key.

5.4.7.4 After Joining

After a node has joined, been authenticated using key establishment, and obtained an authorized link key, it may need to communicate with other nodes on the network using APS layer encryption.

Rather than use key establishment with each node on the network, it would be advantageous to leverage the Trust Center to broker trust with other devices on the network. If two nodes have both obtained link keys with the Trust Center using key establishment, then they both trust the Trust Center. Both nodes will use the Trust Center to request a link key with each other. The trust center will respond to each node individually, sending a randomly generated link key. Each message will be encrypted using the individual nodes' link keys. The Trust Center would not send a link key to either node if one of the nodes has not authenticated using key establishment.[51]

The originating node would start this process by sending a bind request command with APS ack to the key establishment cluster of the destination device. If a bind confirm is received with a status of success, the initiating device will perform a request key of the trust center (for an application link key using the EUI of the other device in the pair). The trust center will then send a link key to each device

49. CCB 1160
50. CCB 1160
51. CCB 1159

using the key transport. If the bind confirm is received with a status other than success, the request key should not be sent to the trust center.

This functionality is optional, however support of this is required for ESI[52] devices acting as trust centers. All devices sending the request key command and the trust center should have a timeout of 5 seconds.

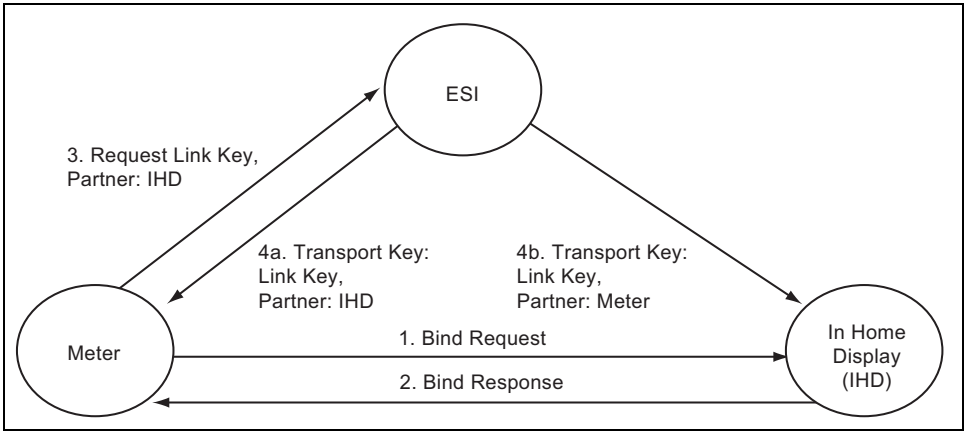

Figure 5.5 Node Communication with Other Nodes on the Network Using APS Layer Encryption

The advantages of using the stack primitives to request keys rather than key establishment are that devices can forego the expensive ECC operations. Small microprocessors have extremely limited resources and requiring full key establishment with all devices where link keys are required is overly burdensome. In addition, ESIs may have other security policies in place (such as node blacklists or certificate revocation lists) that individual nodes do not have knowledge of, or have the resources to keep track of.

Nodes that are not the trust center would not be allowed to initiate key establishment with another device that is not the Trust Center. If a device receives an Initiate Key Establishment Request from a device that is not the Trust Center, and it is not the Trust Center, it shall terminate the key establishment immediately with a status of NO_RESOURCES. This insures that the ESI[53] authenticates all devices with key establishment after joining, and limits the use of key establishment in the network.

Other ESI[54] devices on the network that are not the trust center would have to go through the same procedure as above, contacting the ESI[55] trust center, in order to send/receive messages that require APS layer encryption with another node.

52. CCB 1072
53. CCB 1072

5.4.8 Security Best Practices

5.4.8.1 Out of Band Pre-Configured Link Key Process

This section describes the out of band process for establishing pre-configured Trust Center link keys, the format of the Installation Code required, and the hashing function used to derive the pre-configured link key from the Installation Code.

As portrayed in Figure 5.6, during the manufacturing process a random[56] Installation Code is created for each of the Smart Energy devices. This Installation Code is provided for the device in a manufacturer-specific way (labeling, etc.) and referenced to during installation. The space of installation codes should possess the same randomness properties as a key space. Knowing a set of installation codes should not yield any knowledge of another installation code; and each installation code should be equally probable.[57] The associated Pre-configured Link Key is derived using the hashing function described below and programmed in the device.

Step 1: An Installation Code is created and made available

Step 2: The Pre-configured Link Key is derived from the Installation Code using the Matyas-Meyer-Oseas hash function

Step 3: The Pre-configured Link Key is configured in the device

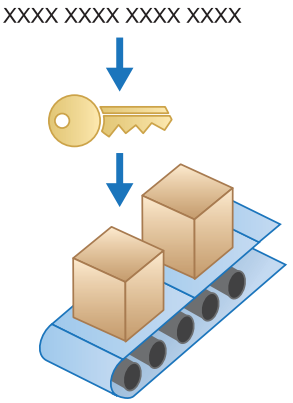

Figure 5.6 Smart Energy Device Installation Code Process

As portrayed in Figure 5.7, during the installation process the initial Trust Center Link Key is derived from the Installation Code and sent via an out of band communication channel to the Trust center (ESI[58]). The Trust center uses this Key as the Trust Center Link Key to subsequently configure the Network Key of the associating device.

54. CCB 1072
55. CCB 1072
56. CCB 1160
57. CCB 1160
58. CCB 1072

Step 1: The Installation Code is sent out of band

Step 2: The Pre-configured Link Key is derived from the Installation Code using the Matyas-Meyer-Oseas hash function

Step 3: The Pre-configured Link Key is sent to the Trust Center using the AMI network

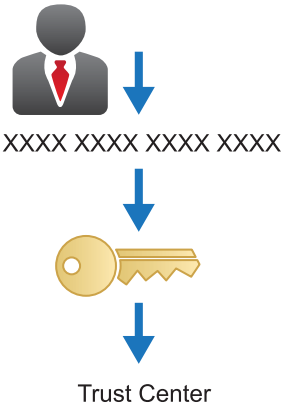

Figure 5.7 Installation Code Use with the Trust Center

5.4.8.1.1 Installation Code Format

The Installation Code consists of a 48, 64, 96, or 128 bit number and a 16 bit CRC (using CCITT CRC standard polynomial X16 + X12 + X5 + 1). When printed or displayed, Installation Codes are represented as multiple groups of 4 hexadecimal digits.

48 Bit example:

Installation Code of “83FE D340 7A93 2B70”
Where values 0x83, 0x FE, 0xD3, 0x40, 0x 7A, and 0x93 are used to calculate the CRC16 with the result returning 0x702B.

Note: *The Octet order of the CRC code in the printed Installation code is Least Significant Octet followed by Most Significant Octet, giving the printed result of “2B70”.*[59]

64 Bit example:

Installation Code of “83FE D340 7A93 9738 C552”
Where values 0x83, 0x FE, 0xD3, 0x40, 0x 7A, 0x93, 0x 97, and 0x38 are used to calculate the CRC16 with the result returning 0x52C5.[60]

96 Bit example:

Installation Code of “83FE D340 7A93 9723 A5C6 39FF 4C12”
Where values 0x83, 0xFE, 0xD3, 0x40, 0x7A, 0x93, 0x97, 0x23, 0xA5, 0xC6, 0x39 and 0xFF are used to calculate the CRC16 with the result returning 0x124C.[61]

59. CCB 980
60. CCB 980
61. CCB 980

128 Bit example:

Installation Code of “83FE D340 7A93 9723 A5C6 39B2 6916 D505 C3B5” Where values 0x83, 0xFE, 0xD3, 0x40, 0x7A, 0x93, 0x97, 0x23, 0xA5, 0xC6, 0x39, 0xB2, 0x69, 0x16, 0xD5, and 0x05 are used to calculate the CRC16 with the result returning 0xB5C3.[62]

5.4.8.1.1.1 CRC Algorithm Information

As stated earlier, the Installation Code CRC calculation is based upon the CRC 16-CCITT algorithm and uses the following parameters:

Length: 16
Polynomial: x16 + x12 + x5 + 1 (0x1021)
Initialization method: Direct
Initialization value: 0xFFFF
Final XOR value: 0xFFFF
Reflected In: True
Reflected Out: True

Open source implementations of the CRC 16-CCITT algorithm are available on the internet at sites like SourceForge and others. The source code is also available for download from the ZigBee document management system [B6].[63]

5.4.8.1.2 Hashing Function

An AES-128 key is derived form the Installation Code using the Matyas-Meyer-Oseas (MMO) hash function (specified in Annex B.6 in ZigBee Document 053474r17, The ZigBee Specification, ZigBee Technical Steering Committee (TSC) with a digest size (hashlen) equal to 128 bits).

Installation Code examples:

- MMO hash applied to the Installation Code “83FE D340 7A93” produces the key “CD4FA064773F46941EC986C09963D1A8”.

Note: *Least significant byte is 0x83 and Most significant byte is 0x93.*

- MMO hash applied to the Installation Code “83FE D340 7A93 9738” produces the key “A833A77434F3BFBD7A7AB97942149287”.

Note: *Least significant byte is 0x83 and Most significant byte is 0x38.*

- MMO hash applied to the Installation Code “83FE D340 7A93 9723 A5C6 39FF” produces the key “58C1828CF7F1C3FE29E7B1024AD84BFA”.

Note: *Least significant byte is 0x83 and Most significant byte is 0xFF.*

62. CCB 980
63. CCB 1060

- MMO hash applied to the Installation Code “83FE D340 7A93 9723 A5C6 39B2 6916 D505” produces the key “66B6900981E1EE3CA4206B6B861C02BB”.

Note: *Least significant byte is 0x83 and Most significant byte is 0x05.8*

5.4.8.1.2.1 MMO Hash Code Example

Open source implementations of the MMO Hash based on the Rijndael implementation are available on the internet at sites like SourceForge and others. The source code is also available for download from the ZigBee document management system [B6].[64]

[65]

5.5 Commissioning

Many, if not all of the devices described in this document, will require some form of commissioning, even if the user or installer doesn't see it. This is because, for example, a load control device needs to be bound to some sort of control device in order to perform its function and, even if the required initializations are done at the factory before the device is installed, the required operations are virtually the same as is the outcome.

The ZigBee Alliance has recognized the importance of commissioning and, in particular, the importance of specifications for network and stack commissioning in a multi-vendor environment. Thus, network and stack commissioning procedures are being designed outside the context of any particular profile, where possible, and grouped under the auspices of the Commissioning Tools Task Group (CTTG). This task group is developing a commissioning framework specification [B2].

5.5.1 Forming the Network (Start-up Sequence)

Smart Energy devices must form their own network or join an existing network. The commissioning framework [B2] discusses some of the relevant issues in this procedure.

It is intended that an installer of a Smart Energy device know if the device is forming a network or joining an existing network.

64. CCB 1060
65. CCB 1159

If a device is forming a network there is no user interaction required since the form process can be completed by the device. However there should be some indication to the user or installer that the network has formed properly. The indication can be implemented in a number of ways including blinking indicator lights, colored indicator lights, arrays of indicator lights, text displays, graphic displays, audible indicators such as buzzers and speakers, or through separate means.

If a device is joining an existing network, it will join the network using the processes outlined in sub-clause 5.4. Permit joining will have been turned on due to either installer action or some backchannel mechanism because of user or installer action. It is recommended there be some indication to the user that the device has joined the network successfully. The indication can be implemented in a number of ways including blinking indicator lights, colored indicator lights, arrays of indicator lights, text displays, graphic displays, audible indicators such as buzzers and speakers, etc.

5.5.2 Support for Commissioning Modes

Three different commissioning modes are discussed in [B2]. They are denoted A, E and S-mode.

As discussed above, Smart Energy devices will either automatically form or join a network based on the processes outlined in sub-clause 5.4.

The pre-installation of start up parameters could be done at manufacturing (which is defined as A mode), by an installer tool at the dispatching warehouse, or on site (which would then be S mode). Devices that support this pre-installation must document the methods used for this preinstallation of parameters.to accomplish this process.

Those devices that will join an existing network must support button pushes or simple documented user interfaces to initiate the joining process. This is in support of E mode commissioning.

5.5.3 Commissioning Documentation Best Practices

To ensure a uniform user experience when commissioning Smart Energy devices, all ZigBee Smart Energy devices are required to provide documentation with their product that explains how to perform device commissioning in using a common language set, i.e., “form network”, “join network”, etc. Please refer to [B2] for further guidance using installation tools and procedures.

5.5.4 Commissioning Procedure for Different Network Types

Depending on the type of network being installed, the commissioning procedures may be slightly different. To ensure interoperability even within these different methods the specific steps are detailed here.

5.5.4.1 Commissioning for Neighborhood Area Network or Sub-metering

Under a neighborhood area network, other meters such as gas or water meters may join electric meters that form a backbone of the network. The process of joining the network is separate from the process for device binding where the device billing information is configured for a particular dwelling unit. It may be desirable to allow the meter to join an adjacent dwelling unit from a network standpoint to ensure proper connectivity. The application level will handle the configuration of the billing information later.

1. There are two methods for joining such a device onto an existing network:
   a. The device is commissioned using a tool with the necessary network and security start up parameters to allow it to rejoin the network as a new device. The device can rejoin any device in the network since it has all the network information.
   b. The network has permit joining turned on by an external tool and the device joins this network and undergoes joining and authentication as any newly joined device.
2. Once joined and authenticated by the security requirements of the existing network, the device is now a member of the neighborhood area network.
3. At the application level, the particular device ID is associated with a particular dwelling unit for billing purposes. This information may be associated at the backend database where the data is collected, or may be sent to the device so it is aware of its association. Note that under this method, devices may route data through devices in adjacent dwelling units that are part of the neighborhood area network.

5.5.4.2 Commissioning for Home Area Network

Under a home area network, the network consists of devices in a particular dwelling unit with one or more co-located metering devices or ESI[66] that provides connectivity to the utility network. Under this scenario, the device within the

66. CCB 1072

home may be installed by a trained installer or by a homeowner. The following steps are completed:

1. The Smart Energy network must be informed of the device that is to be joined. This is done through an out of band means which could include a web login, phone call to a service center, or handheld tool. Using this methodology the existing network is made aware of the device ID and security information appropriate for the device (per the Key Establishment Cluster described in Annex C).
2. The Smart Energy network is put into permit joining ON for a period of time.
3. The installer/homeowner is prompted to press a button or complete a menu sequence that tells the device to attempt to join a network.
4. The device joins the network and is authenticated using the appropriate security mechanisms per the Key Establishment Cluster.
5. An indicator is provided for the installer/homeowner indicating the device has joined a network and authenticated properly or provides information about improper authentication.
6. The device can now operate normally on the network.

5.5.5 ZigBee Smart Energy Joining, Service Discovery, and Device Binding Requirements

Commissioning of a device into a ZigBee Smart Energy network should be easy, reliable, and deterministic. Ideally, a new device could be installed by the home owner or installer communicating the device install code out of band to the coordinator/trust center and then simply powering up the device or manually putting the device into a commissioning (auto-join) state. The device should automatically handle all the steps needed to discover and join the correct PAN and establish relationships with other devices in the HAN without user intervention. As network or HAN conditions change, the devices should be able to adapt automatically without user intervention. ZigBee Smart Energy networks are supposed to last for decades, but once commissioned, devices should require no user interaction in order to remain part of the ZigBee PAN.

Devices that are[67] configured with a Startup Parameter of two (un-commissioned) should automatically begin or make easily available a way to go to Auto-Joining State as described below. (See sub-clause 5.3.1 for the SE Profile Startup Parameter set.)

67. CCB 1382

5.5.5.1 PAN Auto-Joining State

1. When auto-joining state is initiated, a device shall periodically scan all startup set channels for networks that are allowing joining. (See sub-clause 5.3.1 for startup set channel description). A recommended periodic schedule would be:
   a. Immediately when auto-joining state is initiated.
   a. If auto-joining state fails, retry once a minute for the next 15 minutes, jittered by +/- 15 seconds.
   a. If those joining states fail, then retry to join once an hour jittered +/- 30 minutes.
2. To find prospective networks to join, the joining node shall send Beacon Request packets on each channel, dwelling on each channel as specified by the ZigBee Pro specification beacon response window.
3. When a beacon is heard and it has the “Permit Joining” bit set, the device shall attempt to join that PAN. It is up to the implementation of the device to decide if it wants to survey all channels and build a list of joinable PANs before attempting a join procedure, or if it should attempt to join each PAN on a beacon-by-beacon basis. The device shall use its preconfigured link key (derived from a hash of the installation code) to join the targeted SE PAN. Exchanging keys in the clear or with well known preconfigured link keys is not allowed.
4. If the device joins the network but receives a network key that it cannot decrypt, then it has likely joined an incorrect PAN and should back out and try the next joinable PAN. This situation happens most often when the out of band mechanism to communicate the installation key is flawed, or when more than one PAN is allowing joining. It is permissible to try and join the same network again, but not recommended that it be done more than three times in succession. It is expressly not allowed that a device repeat this step more than ten times without backing off to step two and scanning for other networks to join.
5. After the device joins the PAN and is granted ZigBee network key, it must perform service discovery to find a ZigBee key establishment cluster server, then perform ZigBee key establishment in order to get an APS layer link key.
6. If this key establishment fails, it is likely that one side of the exchange is configured with an invalid certificate or with no certificate at all. It is permissible to retry this step multiple times in succession, but it is expressly not allowed that a device repeat this step more than ten times in succession without pausing for a minimum of least fifteen minutes. Since the device was able to get a network key from the Trust Center, the device must have found the correct PAN to join, so there is no need to leave the network. A device that

does not initiate Key Establishment with the Trust Center within a reasonable period of time may be told to leave depending on the network operator's policy.[68]

7 Once key establishment succeeds, the device has joined the correct PAN and shall never leave the PAN without direction from another device in the network (typically an APS *Remove Device* command from an ESI or ZigBee Network Manager) or direction from the user via the device user interface. Example user interfaces could be a text menu or a simple button push sequence. It is strongly recommended that the user interface procedure to get a device to leave the PAN be explicit and difficult to trigger accidentally. Leave commands received over the air should only be followed if the command is an APS encrypted APS *Remove* command. Network layer leave commands should be ignored unless the device is an end device, and the network leave command originated from the parent device.[69]

8 A device that leaves a ZigBee network shall discard its network settings and link key, and revert to its install code, and wait for user input to return or automatically return to auto-join state step one. The device will require the out-of-band registration process to join a new network.

5.5.5.2 Service Discovery State:

1 After the device has successfully performed key establishment, it should use ZigBee Service Discovery mechanisms to discover other devices on the network that have services that match with the device's. This would apply to ZigBee Smart Energy clusters that support asynchronous event commands, like DRLC, Messaging, and Price clusters. For example, a load control device would use ZigBee service discovery to find ESIs that support the load control cluster server. (See sub-clause 5.4.5.1 for more details.)

2 When a matching service is discovered, the device shall use ZigBee device bind mechanisms to send a binding request to the matching device endpoint. It is possible that more than one device with matching services will be discovered. If the device is not an ESI and the ESI are the matching device(s), the device should send binding requests to all ESI with matching services. See the "Multiple ESI Application Guidelines" for more details. Hence a device that wishes to receive unsolicited messages from an ESI on the Messaging Cluster, Price Cluster, DRLC Cluster, shall issue a bind request to the ESI for each cluster it is interested in.

3 A device that sends a binding request is simply announcing itself to an ESI that it desires certain sets of information that the ESI may presently have or may obtain in the future, such as pricing information or DRLC event schedules. The

68. CCB 1286
69. CCB 1286

ESIs that receive bind requests are free to refuse it, but if they refuse the binding request, they must choose another method (an address table for instance) to note the device's interest. Once a device has issued the binding request, it does not need to receive a binding response success. If the device receives a NOT_SUPPORTED (or other non-success code) response to a cluster device bind request, it should still send binding requests for any remaining clusters that it has not sent already.

4 After the device has discovered and bound to matching services, it has now established an application layer relationship with all other relevant devices in the HAN. (See sub-clause 5.7.2 for details of how to deal with multiple time servers and other duplicated services.)That does not mean that the HAN is static and will not acquire new devices, replace devices, or power on devices that were not present during the initial discovery phase. To account for a dynamic HAN, devices shall:

a Repeat the discovery phase on a period of no more than once every three hours and no less than once every 24 hours.

b Repeat the discovery phase after successfully exiting the Rejoin and Recovery Phase (see below).

c Optional - Repeat the discovery phase when a device announce broadcast for a full function device is received. The beginning of the discovery phase should be jittered between 60 and 600 seconds and should be directed only at the device that sent the device announce broadcast.

5.5.5.3 Device Steady State

This is the normal state of the device.

1 A device should make efforts to remain on the correct channel of the PAN and also to keep its network and application keys in sync with the trust center. It is possible that the device has missed a key roll or a channel change due to interference or while it has been powered down or asleep. In order to detect these types of network changes devices shall perform some sort of APS layer message exchange with an ESI on a regular basis. This is to establish that the device can still communicate with the ESI using a current network and APS layer key. This exchange should be performed in accordance with the keep-alive method described in sub-clause 5.4.2.2.3.4.[70] Devices that do not support APS encrypted clusters (Range Extenders for example), do not need to send APS encrypted packets to the Trust Center, but can send network encrypted packets instead.

2 What periodic APS layer message exchange is performed is up to the implementation. Examples would include:

70. CCB 1286

a. Reading a mandatory time cluster attribute (such as *CurrentTime*) on the ESI (recommended). This should work for all ESI.

b. Reading the current consumption attribute on the simple metering cluster (if the ESI supports the simple metering server).

c. Requesting next pricing info from the ESI (if the ESI supports the price cluster server).

3 If the device attempts to perform the periodic message exchange and it fails for any reason, the device should note the failure and retry another exchange later. If after no more than twenty-four hours of retries have failed, the device shall go into the Rejoin and Recovery Phase. It is left to the implementation to decide how many retries should occur within the 24 hour period. It is also permissible for the device to enter the Rejoin and Recovery Phase earlier than 24 hours based on number of failed retries or other factors.

4 4 Sleepy end devices are not required to periodically communicate with an ESI. Instead they should periodically poll their parents and if no parent is found after a suitable period find and rejoin to a new parent. If no parent is found on the original channel, the end device should enter the Rejoin and Recovery phase described below to find a new parent.

5.5.5.4 Rejoin and Recovery State

A device in Rejoin and Recovery Phase is trying to get in sync with its PAN.

1 The device in R&R Phase shall first attempt a ZigBee secure rejoin procedure on its current channel. If the secure rejoin procedure succeeds, the device should revert to its steady state behavior.

2 If the secure rejoin procedure fails, it shall attempt to do a trust center rejoin procedure on its current channel.

3 If the trust center rejoin procedure fails, it may optionally retry steps one and two up to three times.

4 If all attempts to rejoin on the current channel fail, the device shall scan all other channels for its PAN by issuing beacon requests. Note that the PAN ID may have changed and the device shall compare with the extended PAN ID in the beacon and not the short PAN ID.

5 If the device finds an extended PAN ID match in a received beacon, it shall repeat steps one and two on the new channel.

6 If the rejoin (and optional retries) fail on the new channel, the device shall continue scanning all remaining channels for its PAN.

7 If no correct PANs are discovered on any channel, the device shall return to its original channel to wait for the next R&R attempt.

8 If all rejoin attempts on all channels fail, the device shall return to its original channel to wait for the next R&R attempt. This means that the device is back on the original PAN channel, is still a member of the original PAN, (it has not left the network, and has not discarded any PAN information or security keys), and is simply waiting for the rest of the PAN to appear or to time out and begin another R&R attempt.

9 If while waiting for the next R&R attempt, the device receives an APS encrypted message from an ESI and is encrypted with the device's current network and APS layer key, the device shall leave the R&R phase and proceed to the steady state phase.

10 While in the R&R phase, the device shall retry steps 1-8 periodically, at least once every 24 hours. Sleepy end devices may use a longer period. After four failed rejoin attempts, devices should not try to rejoin any faster than once per hour, with a jitter of +/- 30 minutes.

5.5.5.5 ESI Specific Considerations

1 ESI that are not the PAN coordinator, trust center, or network manager shall perform the steady state phase and rejoin and recovery phase as described above.

2 ESI shall support at a minimum, through bindings or other means, at least five separate devices, with enough resources for each device to bind to all of the relevant clusters that the devices may request bindings to. For example, if the ESI supports five smart energy clusters that devices may send binding requests for, the ESI must support twenty five binding relationships, as well as five sets of device ids and security keys

3 It is strongly recommended that ESI operators remove inactive or deprecated devices from the HAN as well as ESI key and binding tables before adding new devices in order to make room for the new device(s). This use case is an example of a device replacement in the HAN.

4 The Trust Center shall never issue an APS *Remove* command without an explicit request from another device on the network or from the head-end network management system.

5 When a new device is registered with the ESI, the ESI may not have enough resources to support it. If the ESI is low on resources, it should notify the installer or ESI administrator (this could be via user interface, or backhaul/ backchannel communication for example.) The ESI shall not automatically remove other devices in order to free up resources for the new device without explicit approval from the installer or ESI administrator.

6 If a device joins the PAN, but does not successfully perform Key Establishment, a trust center may remove the device. This shall only be done after more than 1 hour has elapsed since the device's initial join.[71] This shall be sent directly to the router, or to the parent of an end device. A child that receives a NWK leave from its parent when it does not have an authorized link key (i.e. not performed key establishment successfully) shall not ignore the leave.

7 It is permissible and encouraged that ESI perform its own service discovery procedure after power up and on a periodic basis. The ESI may independently create its own bindings to devices with matching services. This may in some cases establish application layer relationships faster than waiting for devices to request bindings by themselves. As specified in sub-clause 5.4.5.1, an ESI that does not support bindings shall perform its own service discovery.

5.6 Public Pricing

It is required that some ZigBee Smart Energy devices respond to requests for public pricing information from devices that are not on the ZigBee Smart Energy network and do not share the same security settings as the ZigBee Smart Energy devices. Only those devices expected to support and communicate this public pricing information must implement this functionality. Devices that support public pricing must support the price cluster within the ZigBee Smart Energy profile. The data from this ZigBee Smart Energy profile is used as the basis for the public pricing broadcast. In other words, the ZigBee Smart Energy devices that receive pricing information over the Smart Energy network transmits it anonymously and publicly to non-Smart Energy network devices using the anonymous Inter-PAN transmission mechanism outlined in Annex B. Likewise, a Smart Energy device that receives a request for the latest pricing message (formatted as a pricing request from the Price Cluster) will respond with a public and anonymous pricing message.

5.7 Multiple ESI Application Guidelines[72]

5.7.1 Overview

The ZigBee Smart Energy Profile allows for the use of multiple ESIs in a HAN. This feature is desirable from a reliability perspective, plus opens opportunities for vendors to innovate and provide additional services and functionality. Multiple

71. CCB 1286
72. Incremental Release 1

ESIs does not mean multiple Trust Centers, only a single Trust Center is supported in a HAN.

Clients may assume that all SE messages/directives (Demand Response events, price publishing, messaging) are created by the same entity, e.g., utility or energy management entity, or set of coordinated entities. These messages can be sent to devices via one or more transport mechanisms (in the HAN, this means the same message may be sent from multiple ESIs). A message with a specific ID typically will be unique within the system, even though a device may receive this message more than once. However, in a HAN with multiple, uncoordinated commodity service providers [73](e.g., gas vs. water, household electricity vs. PEV electricity), there is a possibility that different, unique events will have conflicting event IDs. Since it is expected that ID conflicts for events occurring at similar points of time will be rare, clients may ignore the issue and always assume that conflicting event IDs are duplicates. More complex clients may choose to better track events by service provider, commodity type, etc.[74]

5.7.2 Device Behavior

5.7.2.1 Service Discovery in Multi ESI Environments

A device should make itself aware of any and all ESIs in a SE HAN using service discovery. It shall[75] perform this service discovery upon joining a network, power up (and network rejoin), and periodically. The typical[76] period is once every 24 hours. A device that discovers an ESI with matching services[77] shall create bindings on the ESI so that the ESI will register the device and send it appropriate SE commands. Devices which do not bind in a multiple ESI network are expected to poll the ESIs. ESIs that are not rediscovered over the period of multiple discovery cycles may be forgotten by the device.

A device that discovers more than one ESI should determine a single ESI as an authoritative time source. To do so, it should use the Time cluster Master, Synchronized, and Superseding bits.[78]

A Time server with the Superseding bit set will always take precedence over a Time server without that bit set, including ones that have the Master bit set. A new ESI going into a faulty installation can set the Superseding bit and take over the network's Time synchronization. However, it is not required for SE 1.x to have

73. CCB 1352
74. CCB 1350
75. CCB 1350
76. CCB 1350
77. CCB 1383
78. CCB 1352

this bit set if the new ESI does not want to forcefully take over the Time server role. This bit is set independently of the other three *TimeStatus* bits (Master, Synchronized, MasterZoneDst).

5.7.2.2 Determining the Most Authoritative Time Source

Devices shall synchronize to a Time server with the highest rank according to the following rules, listed in order of precedence:

1. A server with the Superseding and Master bits set shall be chosen over a server with only the Master bit set.
2. A server with the Master bit set shall be chosen over a server without the bit set.
3. 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).
4. A Time server with neither the Master nor Synchronized bits set should not be chosen as the network time server.

5.7.2.3 Periodic Time Source Checking During Normal Operation[79]

During normal operation (the most authoritative time source is found and it has valid time), clients periodically repeat the time source scan to pick up new, more authoritative time sources, as per the following rules:

1. Non-sleepy clients shall locate the most authoritative time source at least once every 24 hours.
2. Sleepy devices should locate the most authoritative time source at least once every 24 hours.
3. Clients shall scan for time sources after rebooting, and after joining or rejoining the network.

5.7.2.4 Invalid Time and Interim Time Sources[80]

Although the rules above are used to find the most authoritative time source for the network, there are conditions where what would normally be the most authoritative source is temporarily unable to provide valid time. In this situation, regardless of whether it is encountered as part of the original time discovery or the periodic rediscovery, devices obey the following rules:

79. CCB 1350
80. CCB 1350

1. If a server is temporarily unable to provide valid UTC, it shall report all time attributes (e.g., UTC, local time) as 0xFFFFFFFF. It should leave the Superseding, Master, and Synchronized bits set as if it did have valid time.
2. If the most authoritative time source for the network has invalid time, clients should temporarily use the Time server of next highest rank, but shall periodically look for the more authoritative server(s) to obtain valid time. Non-sleepy devices shall check at least once every 15 minutes. Sleepy devices may check as often as their power budget will allow.
3. When a more authoritative time source with valid time is found, clients shall immediately switch to using that source's time basis.

5.7.2.5 Handling SE Commands from Multiple ESIs[81]

When a device creates bindings on multiple ESIs, it may receive SE commands from those ESIs. Simple device logic such as assuming all commands came from the PAN coordinator is not appropriate. The following rules describe the desirable device behavior.

1. When a device receives an event (Demand Response, Price, Messaging) any time reference in the message should be viewed in context with the time reference of the most authoritative ESI time server.
2. When a device receives duplicate events (same event ID) from multiple ESIs, it shall send an event response to each ESI. Future duplicate events from the same ESI(s) shall be either “ignored” by sending no response at all or with a default response containing a success status code.[82]
3. Conflicting events with the same event ID from different ESIs will be resolved by the device in the same manner as if they came from a single ESI.
4. When a device has an asynchronous follow up event response it should send the response to the ESIs that created the condition. If the event was received from more than one ESI, the device shall send the asynchronous event response to all ESIs from which it received the event.[83]

5.7.2.6 Handling Multiple Uncoordinated Back-end Systems[84]

When multiple, uncoordinated service providers deploy ESIs in a HAN, it is possible that different back-ends/ESIs will have a different notion of time. However, only one ESI will be the authoritative time source for the HAN. In this

81. CCB 1350
82. Toronto 1.1.1 certification event 8/26/2011
83. Toronto 1.1.1 certification event 8/26/2011
84. CCB 1350

scenario, an ESI may require a mechanism to ensure that its events are executed on its time basis, even if it is not the authoritative time source. ESIs that require this behavior may make use of the following application guidelines:

1. An ESI may implement the Time cluster client and determine the most authoritative time source for the HAN, using the rules defined above.
2. If the ESI is not the authoritative time source, it may synchronize its clock, or apply differentials to the start time of its events, to ensure that clients execute those events on the intended schedule.
3. If the more authoritative time server disappears and is not seen for 24 hours, clients may assume that the server has left the network, and resume normal operation using the most authoritative time server (with valid time) that remains.

5.8 Other Smart Energy Profile Requirements and Best Practices

5.8.1 Preferred Channel Usage

When forming a new network, or scanning to join a network, Smart Energy devices should do channel scans using the following preferred channels before scanning the rest of the channels in order to avoid the most commonly used WiFi channels. This is to improve the user experience during installation (quicker joining) and possibly improve bandwidth (on average).

Preferred 2.4 GHz Channels - 11, 14, 15, 19, 20, 24, 25

Preferred 900MHz Channels – Use all available for ZigBee.

5.8.2 Broadcast Policy

Except for public pricing, broadcasts are strongly discouraged for Smart Energy devices. Devices are limited to a maximum broadcast frequency of one broadcast per second and strongly encouraged to exercise broadcasts much less frequently.

5.8.3 Frequency Agility

Frequency Agility would only be officially exercised in a network by a system controller, or higher functioning device (ESI[85], aggregator, installation tool, etc…).

85. CCB 1072

Devices may support frequency agility hooks to be commanded to “go to channel X”. Devices that do not support frequency agility may implement either the NWK rejoin or orphan join feature to find a network that has changed channels.

5.8.4 Key Updates

Smart Energy devices are only required to support ZigBee “residential mode” security or ZigBee PRO “standard mode” with the required use of link keys. All link key updates shall use the Key Establishment Cluster. Sleeping devices that miss key updates can request a new key using the existing link key so there is no problem with sleeping devices missing key updates.

DELETED SECTION 5.8.5[86]

5.9 Coexistence and Interoperability with HA Devices

It is desirable to allow interoperability of HA and Smart Energy devices where practical. However, it is undesirable to publicly share keys during the joining process or share private information over a less secure network. HA devices that only provide functionality for receiving network keys in the clear during a join process cannot be used in a Smart Energy network. These devices can receive public pricing information as described above. HA devices may also be extended with Smart Energy clusters providing they support the use of Link Keys and the Smart Energy security models. If so, they can be certified as HA and Smart Energy capable allowing those devices to operate either in an HA network or a Smart Energy network.

5.10 Device Descriptions

Device descriptions specified in this profile are summarized in Table 5.14 along with their respective Device IDs. The devices are organized according to the end application areas they address. A product that conforms to this specification shall implement at least one of these device descriptions and shall also include the device descriptions corresponding to all applications implemented on the product where a standard device description is specified in this profile. For example, if a product implements both a thermostat and an In-Home[87] Display, then the thermostat and In-Home[88] Display device descriptions must both be supported.

86. CCB 1289
87. Incremental Release 1/CCB 1570
88. Incremental Release 1/CCB 1570

This list will be added to in future versions of the profile as new clusters are developed to meet the needs of manufacturers. The reserved values shall not be used until the profile defines them. Manufacturer-specific device descriptions shall reside on a separate endpoint and use a private profile ID.

Table 5.14 Devices Specified in the Smart Energy Profile

	Device	Device ID[a]
Generic	Range Extender	0x0008
Smart Energy	Energy Service Interface[b]	0x0500
	Metering Device	0x0501
	In-Home[c] Display	0x0502
	Programmable Communicating Thermostat (PCT)	0x0503
	Load Control Device	0x0504
	Smart Appliance	0x0505
	Prepayment Terminal	0x0506
	Physical Device[d]	0x0507
	Reserved	0x0508 – 0x5FF

a. CCB 1486

b. Incremental Release 1

c. Incremental Release 1/CCB 1570

d. CCB 1486

5.11 ZigBee Cluster Library (ZCL)

This profile utilizes some of the clusters specified in the ZigBee Cluster Library. The implementation details for each cluster are given in the ZCL specifications. Further specification and clarification is given in this profile where necessary.

The ZCL provides a mechanism for clusters to report changes to the value of various attributes. It also provides commands to configure the reporting parameters. Products shall support the attribute reporting mechanism for

supported attributes as specified in the ZCL. The minimum reporting interval specified in the ZCL [B1] shall be set to a value greater than or equal to 0x0001. The maximum reporting interval should be set to 0x0000 by default, and if it is set to a non-zero value it shall be set to a value greater than or equal to 0x003C and greater than the value of the minimum reporting interval. These settings will restrict the attributes from being reported more often than once every second if the attribute is changing quickly and at least once every minute if the attribute does not change for a long time. It is recommended that the minimum reporting interval be set to one minute and the maximum reporting interval be set to a much greater value to avoid unnecessary traffic.

Devices shall use the ZCL default response error handing. Typical examples of this are:

- When receiving commands that don't have data collected such as Get Scheduled Events, Get Current Price, Get Scheduled Prices, Get Block Period(s), and Get Last Message, devices shall respond using the ZCL default response with a status code of NOT_FOUND.
- When receiving requests for unsupported commands, devices shall respond using the ZCL default response with a status code of UNSUP_CLUSTER_COMMAND.
- When receiving malformed commands, devices shall respond using the ZCL default response with a status code of MALFORMED_COMMAND.
- When receiving requests for accessing unsupported attributes, devices shall respond using the ZCL default response with a status code of UNSUPPORTED_ATTRIBUTE.

Please refer to [B1] for additional status codes support in the ZCL default response.

5.12 Cluster List and IDs

The clusters used in this profile are listed in Table 5.15. The clusters are listed according to the functional domain they belong to in the ZCL and indicate the additional new Smart Energy clusters. The existing corresponding ZCL General cluster identifiers can be found in the ZCL [B1].

The functionality made available by all supported clusters shall be that given in their ZCL specifications except where a device description in this profile includes further specification, clarification or restriction as needed for a particular device.

Most clusters include optional attributes. The application designer must be aware that optional attributes might not be implemented on a particular device. All

Smart Energy devices must discover and deal with unsupported attributes on other devices.

It is expected that clusters will continue to be developed in the ZCL that will be useful in this profile. In many cases, new clusters will be organized into new device descriptions that are separate from those currently defined. There may also be situations where it makes sense to add clusters as optional elements of existing device descriptions.

Manufacturer-specific clusters may be added to any device description in this profile as long as they follow the specifications given in the ZCL [B1].

Table 5.15 Clusters Used in the Smart Energy Profile

Functional Domain	Cluster Name	Cluster ID
General	Basic	0x0000
General	Identify	0x0003
General	Alarms	0x0009
General	Time	0x000A
General	Commissioning	0x0015
General	Power Configuration	0x0001
General	Key Establishment	0x0800
Smart Energy	Price	0x0700
Smart Energy	Demand Response and Load Control	0x0701
Smart Energy	Metering[a]	0x0702
Smart Energy	Messaging[b]	0x0703
Smart Energy	Smart Energy Tunneling (Complex Metering)	0x0704
Smart Energy	Prepayment	0x0705

a. CCB 940

b. Incremental Release 1

5.12.1 ZCL General Clusters

Except for the Key Establishment Cluster, which is covered in Annex C, please refer to the ZCL Cluster Specification [B1] for the General Cluster descriptions.

5.12.1.1 ZCL Time Cluster and Time Synchronization

The Smart Energy profile requires time synchronization between devices to properly support the coordination of Demand Response/Load Control events,

Price changes, and the collection of metered data. In order to simplify the understanding of time, the Smart Energy profile will leverage UTC as the common time base. To this end a new ZCL attribute data type, UTCTime is included and its definition can be found in Annex A.

It is desired for the processes for synchronizing time to be as network friendly as possible to eliminate excessive traffic. To support this, time accuracy on Client devices shall be within +/-1 minute of the server device (ESI[89]) per 24 hour period. The Client devices shall design a clock accuracy that never requires more than one time synchronization event per 24 hour period. The exception to this is when devices need to rejoin or re-register on the network. Again, the desire is to keep time synchronization traffic to a minimum.

Further, implementers must be aware that network communication delays will cause minor differences in time between devices. The Smart Energy profile expectations are that this will be a minor issue given the use cases it's fulfilling. It will not nor does it recommend implementers develop an NTP or equivalent scheme to compensate for network delays. These methods are viewed as having the potential to cause excessive network communications.

89. CCB 1072

This page intentionally blank

CHAPTER

6

DEVICE SPECIFICATIONS

6.1 Common Clusters

Support for certain clusters is required on all products supporting this profile. At least one instance of the clusters shown in Table 6.1 shall exist on a product supporting this profile.[90] Individual device descriptions may place further restrictions on support of the optional clusters shown here. ZCL clusters not listed may be implemented on the Smart Energy endpoint. Manufacturers may extend the SE profile as specified in the ZCL specification [B1].[91]

Table 6.1 Clusters Common to All Devices

Server Side[a]	**Client Side**[b]
Mandatory	
Basic	*None*
Key Establishment	Key Establishment
Optional	
Power Configuration	None
Commissioning	Commissioning
Identify	*None*[c]
OTA Upgrade[d]	OTA Upgrade[e]

a. CCB 1486

b. CCB 1486

c. CCB 966

90. CCB 1486
91. Incremental Release 1

d. Incremental Release 1

e. Incremental Release 1

6.1.1 Optional Support for Clusters with Reporting Capability

Some clusters support the ability to report changes to the value of particular attributes. These reports are typically received by the client side of the cluster. All devices in this profile may support any cluster that receives attribute reports.

6.1.2 Manufacturer-Specific Clusters

The ZCL provides a range of cluster IDs that are reserved for manufacturer-specific clusters. Manufacturer-specific clusters that conform to the requirements given in the ZCL may be added to any device description specified in this profile.

6.1.3 Cluster Usage Restrictions

None.

6.1.4 Identify Cluster Best Practices

To help aid in locating devices, it's strongly recommended that all devices utilize the Identify Cluster and a visual or audible indicator. In situations in which a device can't supply a visual or audible indicator, the device should include a visible label with the appropriate information to help identify the device.[92]

6.1.5 Inter-PAN Communication[93]

Inter-PAN access to Smart Energy devices shall be limited to specific clusters and commands. Please refer to Annex B for further details.

6.2 Feature and Function Description

Each device must support a certain set of features and functions. Table 6.2 below is used to specify the mandatory and optional features and functions for Smart Energy devices. This chapter contains a description of what must be supported if

92. CCB 966

93. CCB 1486

the feature or function is supported by the device. The mandatory or optional configuration for each device is described in the upcoming chapters:

Table 6.2 Common Features and Functions Configuration for a Smart Energy Device

Device Type/ Feature or function	Join (end devices and routers only)	Form Network (coordinator only)	Restore to Factory Fresh Settings	Pair Devices – (End Device Bind Request)	Bind Manager – (End Device Bind Response - Coordinator only)	Enable Identify Mode	Allow Smart Energy devices to join the Network (routers and coordinators only)
Mandatory/ Optional	M	M	M	O[a]	M	O[b]	M
Device Type/ Feature or function	Service discovery (Match Descriptor Request)	ZDP Bind Response	ZDP Unbind Response	End Device Annce/ device annce	Service Discovery response (Match Descriptor Response)	High Security Supported (ZigBee PRO only)	Inter-PAN Communication
Mandatory/ Optional	O[c]	M	M	M	M	N/A	O

a. CCB 967

b. CCB 967

c. CCB 967

Join (End Devices and Routers):

As described in sub-clauses 5.4 and 5.5.

Form Network (Coordinator):

As described in sub-clauses 5.4 and 5.5.

Allow Others to Join Network (Router and Coordinator Only):

As described in sub-clauses 5.4 and 5.5.

Restore to Factory Fresh Settings:

The Device shall provide a way for the restore Factory Settings.

Pair Devices (End Device Bind Request):

Whenever possible, the device should provide a way for the user to issue an End Device Bind Request.[94]

Bind Manager (End Device Bind Response – Coordinator only):

The coordinator device shall be capable of issuing an End Device Bind Response.

Enable Identify Mode:

Whenever possible, the device should provide a way for the user to enable Identify for 60 seconds.[95]

Service Discovery (Match Descriptor Request):

Whenever possible, the device should provide a way for device to send a match descriptor request, receive match descriptor responses and utilize them for commissioning the device.[96]

ZDP Bind Response:

The device shall be able to receive a ZDP Bind Request and respond correctly with a ZDP Bind Response.

ZDP Unbind Response:

The device shall be able to receive a ZDP Unbind Request and respond correctly with a ZDP Unbind Response.

End Device Annce/Device Annce:

The device shall Send End Device Annce / Send Device upon joining and re-joining a network.

Service Discovery Response:

The Device shall be able to receive a Match descriptor request, and respond with a match descriptor response correctly.

Allow Smart Energy Devices to Join the Network:

The Device shall allow other Smart Energy devices to join the network.

High Security Supported: No

Inter-PAN Communication:

The device may support Inter-PAN Communications as described in Annex B

94. CCB 967
95. CCB 967
96. CCB 967

6.3 Smart Energy Devices

A physical device may support one or more logical Smart Energy devices. The supported clusters of a logical Smart Energy device shall reside on a single endpoint, with the exception of the common clusters listed in Table 6.1, which may reside on a separate endpoint using the Physical Device identifier.[97] Each logical Smart Energy device on a single physical device shall reside on its own separate endpoint.[98]

SE devices shall use the device and service discovery mechanisms specified in the ZigBee specification [B3] to find the services required. Devices shall support discovery of single and multiple endpoints on a single physical device.[99] In the case where multiple devices of the same type are discovered, SE cluster attributes should be read to determine the type of service provided. For example, if multiple Metering devices are discovered the *MeteringDeviceType* attribute provides a label for identifying the type of metering device present.[100] Similarly, if multiple ESIs are found the *CommodityType* attribute shall be read to determine the fuel type of that ESI.[101]

6.3.1 Energy Service Interface[102]

The Energy Service Interface[103] connects the energy supply company communication network to the metering and energy management devices within the home. It routes messages to and from the relevant end points. It may be installed within a meter, thermostat, or In-Home[104] Display, or may be a standalone device, and it will contain another non-ZigBee communication module (e.g. power-line carrier, RF, GPRS, broadband Internet connection).

6.3.1.1 Supported Clusters

In addition to those specified in Table 6.1, the Energy Service Interface[105] device shall support the clusters listed in Table 6.3. If a SE[106] cluster is not listed as

97. CCB 1486
98. CCB 1265
99. CCB 1265
100. Incremental Release 1
101. CCB 1383
102. CCB 1072
103. CCB 1072
104. Incremental Release 1/CCB 1570
105. CCB 1072
106. Incremental Release 1

mandatory or optional in the following table or in the common table, then that cluster shall be prohibited on an ESI[107] device endpoint.

Table 6.3 Clusters Supported by the Energy Service Interface[a]

Server Side	Client Side
Mandatory	
Messaging[b]	
Price	
Demand Response/Load Control	
Time	
Optional[c]	
	Price
Metering[d]	Metering[e]
Prepayment	Prepayment
	Time[f]
Alarms[g]	
Tunneling and Generic Tunneling[h]	Tunneling and Generic Tunneling[i]

a. CCB 1072
b. Incremental Release 1
c. Incremental Release 1
d. CCB 940
e. CCB 940
f. CCB 1349
g. CCB 1486
h. CCB 1486
i. CCB 1486

6.3.1.2 Supported Features and Functions

The Energy Service Interface[108] device shall have the features and functions listed in Table 6.2.

107. CCB 1072
108. CCB 1072

6.3.2 Metering Device

The Metering end device is a meter (electricity, gas, water, heat, etc.) that is fitted with a ZigBee device. Depending on what is being metered, the device may be capable of immediate (requested) reads or it will autonomously send readings periodically. A Metering end device may also be capable of communicating certain status indicators (e.g. battery low, tamper detected).

6.3.2.1 Supported Clusters

In addition to those specified in Table 6.1, the Metering Device shall support the clusters listed in Table 6.4. If a SE[109] cluster is not listed as mandatory or optional in the following table or in the common table, then that cluster shall be prohibited on a Metering device endpoint.

Table 6.4 Clusters Supported by the Metering Device

Server Side	Client Side
Mandatory	
Metering[a]	
Optional[b]	
	Time
Prepayment[c]	
	Price
	Messaging
Alarms[d]	
Tunneling and Generic Tunneling[e]	Tunneling and Generic Tunneling[f]

a. CCB 940
b. Incremental Release 1
c. Incremental Release 1
d. CCB 1486
e. CCB 1486
f. CCB 1486

6.3.2.2 Supported Features and Functions

The Metering Device shall have the features and functions listed in Table 6.2.

109. Incremental Release 1

6.3.3 In-Home[110] Display Device

The In-Home[111] Display device will relay energy consumption data to the user by way of a graphical or text display. The display may or may not be an interactive device. At a minimum at least one of the following should be displayed: current energy usage, a history over selectable periods, pricing information, or text messages. As an interactive device, it can be used for returning simple messages for interpretation by the recipient (e.g. “Button A was pressed”).

The display may also show critical pricing information to advise the customer when peaks are due to occur so that they can take appropriate action.

6.3.3.1 Supported Clusters

In addition to those specified in Table 6.1, the In-Home[112] Display device shall support the clusters listed in Table 6.5. If a SE[113] cluster is not listed as mandatory or optional in the following table or in the common table, then that cluster shall be prohibited on an In-Home[114] Display device endpoint.

Table 6.5 Clusters Supported by the In-Home[a] Display Device

Server Side	Client Side
Mandatory	
Optional	
	Demand Response and Load Control
	Time
	Prepayment
	Price
	Metering[b]
	Messaging[c]
Alarms[d]	
Tunneling and Generic Tunneling[e]	Tunneling and Generic Tunneling[f]

a. Incremental Release 1/CCB 1570

110. Incremental Release 1/CCB 1570
111. Incremental Release 1/CCB 1570
112. Incremental Release 1/CCB 1570
113. Incremental Release 1
114. Incremental Release 1/CCB 1570

b. CCB 940
c. Incremental Release 1
d. CCB 1486
e. CCB 1486
f. CCB 1486

An In-Home Display shall implement at least one of the optional client clusters listed.[115, 116, 117]

6.3.3.2 Supported Features and Functions

The In-Home[118] Display device shall have the features and functions listed in Table 6.2.

6.3.4 Programmable Communicating Thermostat (PCT) Device

The PCT device shall provide the capability to control the premises heating and cooling systems.

6.3.4.1 Supported Clusters

In addition to those specified in Table 6.1, the PCT device shall support the clusters listed in Table 6.6. If a SE[119] cluster is not listed as mandatory or optional in the following table or in the common table, then that cluster shall be prohibited on a PCT device endpoint.

Table 6.6 Clusters Supported by the PCT

Server Side	Client Side
Mandatory	
	Demand Response and Load Control
	Time
Optional	
	Prepayment

115. CCB 1283
116. Incremental Release 1
117. CCB 1564
118. Incremental Release 1/CCB 1570
119. Incremental Release 1

Table 6.6 Clusters Supported by the PCT (Continued)

Server Side	Client Side
	Price
	Metering[a]
	Messaging[b]
Alarms[c]	
Tunneling and Generic Tunneling[d]	Tunneling and Generic Tunneling[e]

a. CCB 940
b. Incremental Release 1
c. CCB 1486
d. CCB 1486
e. CCB 1486

6.3.4.2 Supported Features and Functions

The PCT device shall have the features and functions listed in Table 6.2.

6.3.5 Load Control Device

The Load Control device is capable of receiving Demand Response and Load Control events to manage consumption on a range of devices. Example devices are water heaters, exterior lighting, and pool pumps.

6.3.5.1 Supported Clusters

In addition to those specified in Table 6.1, the Load Control device shall support the clusters listed in Table 6.7.

Table 6.7 Clusters Supported by the Load Control Device

Server Side	Client Side
Mandatory	
	Demand Response and Load Control
	Time
Optional	
	Price
Alarms[a]	
Tunneling and Generic Tunneling[b]	Tunneling and Generic Tunneling[c]

a. CCB 1486
b. CCB 1486
c. CCB 1486

6.3.5.2 Supported Features and Functions

The Load Control Device shall support the features and functions listed in Table 6.2.

6.3.6 Range Extender Device

The Range Extender is a simple device that acts as a router for other devices. The Range Extender device shall not be a ZigBee end device. A product that implements the Range Extender device shall not implement any other devices defined in this profile. This device shall only be used if the product is not intended to have any other application, or if a private application is implemented that has not been addressed by this profile.

6.3.6.1 Supported Clusters

The Range Extender device shall support the mandatory common clusters listed in Table 6.1.[120]

6.3.6.2 Supported Features and Functions

The Range Extender device shall have the features and functions listed in Table 6.2.

6.3.7 Smart Appliance Device

Smart Appliance devices on the ZigBee network can participate in energy management activities. Examples of these are when Utilities initiate a demand response or pricing event, or the appliance actively informs customers via in-home displays of when or how energy is being used. In the latter case, scenarios include:

- Washer switching to cold water during periods of higher energy costs.
- Washer/Dryer/Oven/Hot Water Heater reporting cycle status.
- Over temperature conditions in Freezers and Refrigerators.

120.CCB 1030

6.3.7.1 Supported Clusters

In addition to those specified in Table 6.1 the Smart Appliance device shall support the clusters listed in Table 6.8. If a SE[121] cluster is not listed as mandatory or optional in the following table or in the common table, then that cluster shall be prohibited on a Smart Appliance device endpoint.[122]

Table 6.8 Clusters Supported by the Smart Appliance Device

Server Side	Client Side
Mandatory	
	Price
	Time
Optional	
	Demand Response and Load Control
	Messaging[a]
Alarms[b]	
Tunneling and Generic Tunneling[c]	Tunneling and Generic Tunneling[d]

a. Incremental Release 1

b. CCB 1486

c. CCB 1486

d. CCB 1486

6.3.7.2 Supported Features and Functions

The Smart Appliance device shall have the features and functions listed in Table 6.2.

6.3.8 Prepayment Terminal Device[123]

The Prepayment Terminal device will allow utility customers or other users (e.g. sub-metered tenants) to pay for consumption in discrete increments rather than establishing a traditional billing agreement. The Prepayment Terminal device will accept payment (e.g. credit card, code entry), display remaining balances, and alert the user of a balance approaching zero, and may perform some or all of the other functions described in sub-clause 6.3.3 “In-Home Display Device”.[124]

121. Incremental Release 1
122. Incremental Release 1
123. Incremental Release 1

6.3.8.1 Supported Clusters

In addition to those specified in Table 6.1, the Prepayment Terminal device shall support the clusters listed in Table 6.9. If a SE[125] cluster is not listed as mandatory or optional in the following table or in the common table, then that cluster shall be prohibited on a Prepayment Terminal device endpoint.

Table 6.9 Clusters Supported by the Prepayment Terminal Device

Server Side	Client Side
Mandatory	
	Price
	Time
Prepayment	Prepayment
Optional	
	Demand Response and Load Control
	Metering[a]
	Messaging[b]
Alarms[c]	
Tunneling and Generic Tunneling[d]	Tunneling and Generic Tunneling[e]

a. CCB 940

b. Incremental Release 1

c. CCB 1486

d. CCB 1486

e. CCB 1486

6.3.8.2 Supported Features and Functions

The Prepayment Terminal device shall have the features and functions listed in Table 6.2.

6.3.9 Physical Device[126]

The Physical Device type will identify a supplemental (or sole) endpoint on which the clusters related to a physical product may reside. The endpoint shall not

124. Incremental Release 1/ CCB 1570
125. Incremental Release 1
126. CCB 1486

contain any cluster related to any individual logical SE device on the physical product. A product is allowed to have a Physical Device as its sole SE endpoint. A Physical Device must be capable of providing other SE device endpoints to be a certified SE product.

6.3.9.1 Supported Clusters[127]

The Physical Device may only support the common clusters listed in Table 6.1.

6.3.9.2 Supported Features and Functions[128]

The Physical Device shall have the features and functions listed in Table 6.2.

127.CCB 1486
128.CCB 1486

ANNEX

A

CANDIDATE ZCL MATERIAL FOR USE WITH THIS PROFILE

The candidate material in this annex, when approved, will be merged into the Foundation document of the ZigBee Cluster Library (ZCL) by the Cluster Library Development Board.

A.1 New Data Types

This section defines new ZCL data types needed for interoperability with Smart Energy-based ZigBee devices and functions.

A.2 Definition of New Types

The following material in this subsection is proposed for inclusion in the Data Types section (section 8.2) of the Foundation document [B1].

A.2.1 New Time Data Type

The new Time data type being requested is listed in Table A.1.

Table A.1 Additional Time Cluster Data Type

Type Class	Data Type ID	Data Type	Length of Data (Octets)	Invalid Number	Analog / Discrete
Time	0xe2	UTCTime	4	0xffffffff	A
	0xe3 – 0xe7	Reserved	-	-	-

A.2.1.1 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. It reflects and defines the data type used in the ZCL Time Server attribute labeled as *Time*. The value that represents an invalid value of this type is 0xffffffffff.

A.2.2 New Unsigned Integer Data Type

The new Unsigned Integers data types being requested are listed in Table A.2.

Table A.2 New Unsigned Integer Data Types

Type Class	Data Type ID	Data Type	Length of Data (Octets)	Invalid Number	Analog / Discrete
Unsigned Integer	0x24	Unsigned 40 Bit Integer	5	0xffffffffff	A
	0x25	Unsigned 48 Bit Integer	6	0xffffffffffff	A
	0x26 – 0x27	Reserved	-	-	-

A.2.2.1 Unsigned 40 Bit Integer

This type represents an unsigned integer with a decimal range of 0 to 2^{40}-1. The value that represents an invalid value of this type is 0xffffffffff.

A.2.2.2 Unsigned 48 Bit Integer

This type represents an unsigned integer with a decimal range of 0 to 2^{48}-1. The value that represents an invalid value of this type is 0xffffffffffff.

ANNEX

B

INTER-PAN TRANSMISSION MECHANISM

B.1 Scope and Purpose

This annex defines a mechanism whereby ZigBee devices can perform limited, insecure, and possibly anonymous exchanges of information with devices in their local neighborhood without having to form or join the same ZigBee network. The mandate for this feature comes from the Energy Management / Smart Energy market requirement to send pricing information to very low cost devices. The particular data exchange required by the Smart Energy Application Profile is the request for anonymous public energy pricing information. The typical example is the extremely low cost “Refrigerator Magnet” device that simply informs customers of current energy costs through some visual method (LCD, LED's, etc.).

The intended destination for the mechanism described here is not the ZigBee specification [B7], but the relevant application profile documents for applications that make use of the feature – in particular, the Smart Energy Profile Specification.

The material used to create Annex B is derived from [B5].

B.2 General Description

B.2.1 What Inter-PAN Transmission Does

A schematic view of the how inter-PAN transmission in a ZigBee context works is shown in Figure B.1.

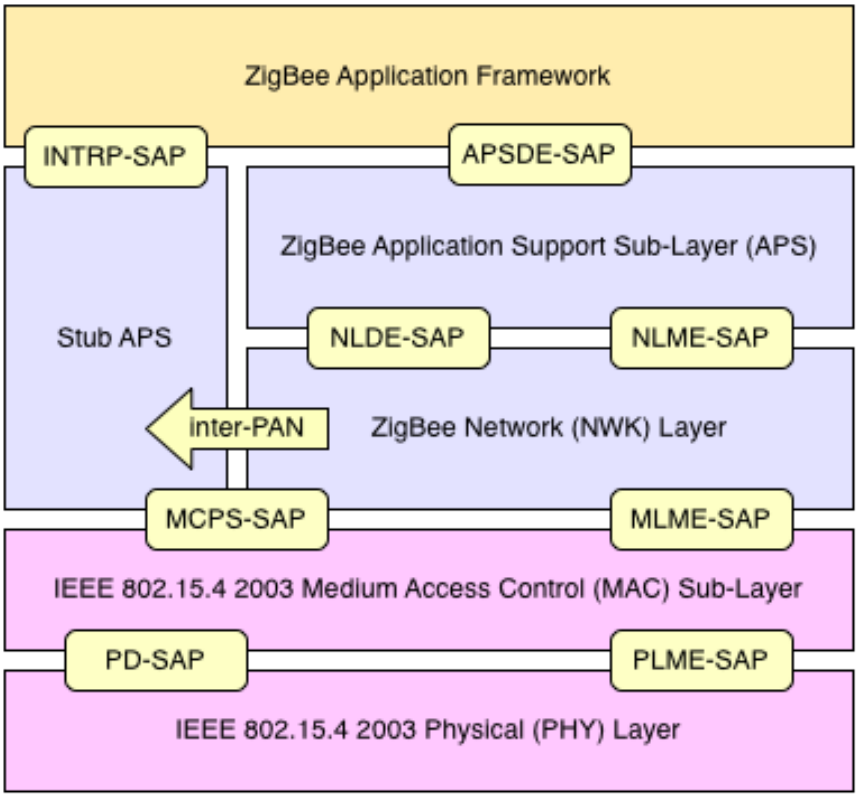

Figure B.1 ZigBee Stack with Stub APS

Inter-PAN data exchanges are handled by a special “stub” of the Application Support Sub-Layer, which is accessible through a special Service Access Point (SAP), the INTRP-SAP, parallel to the normal APSDE-SAP. The stub APS performs just enough processing to pass application data frames to the MAC for transmission and to pass inter-PAN application frames from the MAC to the application on receipt.

The Inter-Pan data exchange architecture does not support simultaneous execution by multiple application entities. Within a device, only one application entity shall use the Inter-Pan communications mechanisms.

B.3 Service Specification

The INTRP-SAP is a data service comprising three primitives.

- INTRP-DATA.request - Provides a mechanism for a sending device to request transmission of an Inter-Pan message.

- INTRP-DATA.confirm - Provides a mechanism for a sending device to understand the status of a previous request to send an Inter-Pan message.
- INTRP-DATA.indication - Provides a mechanism for a identifying and conveying an Inter-Pan message received from a sending device.

B.3.1 The INTRP-DATA.request Primitive

The INTRP-DATA.request primitive allows an application entity to request data transmission via the stub APS.

B.3.1.1 Semantics of the Service Primitive

The primitive interface is as follows:

INTRP-DATA.request	{
	SrcAddrMode
	DstAddrMode
	DstPANId
	DstAddress
	ProfileId
	ClusterId
	ASDULength
	ASDU
	ASDUHandle
	}

Parameters of the primitive appear in Table B.1.

Table B.1 Parameters of the INTRP-DATA.request

Name	Type	Valid Range	Description
SrcAddrMode	Integer	0x03	The addressing mode for the source address used in this primitive. This parameter shall only reference the use of the 64-bit extended address: 0x03 = 64-bit extended address
DstAddrMode	Integer	0x01 – 0x03	The addressing mode for the destination address used in this primitive. This parameter can take one of the values from the following list: 0x01 = 16-bit group address 0x02 = 16-bit NWK address, normally the broadcast address 0xffff 0x03 = 64-bit extended address
DstPANID	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the entity or entities to which the ASDU is being transferred or the broadcast PANId 0xffff.
DstAddress	16-bit or 64-bit address	As specified by the AddrMode parameter	The address of the entity or entities to which the ASDU is being transferred.
ProfileId	Integer	0x0000 – 0xffff	The identifier of the application profile for which this frame is intended.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the cluster, within the profile specified by the ProfileId parameter, which defines the application semantics of the ASDU.
ASDULength	Integer	0x00 – (*aMaxMACFrameSize* - 9)	The number of octets in the ASDU to be transmitted.
ASDU	Set of octets	-	The set of octets forming the ASDU to be transmitted.
ASDUHandle	Integer	0x00 – 0xff	An integer handle associated with the ASDU to be transmitted.

B.3.1.2 When Generated

This primitive is generated by the local application entity when it wishes to address a frame to one or more peer application entities residing on neighboring devices with which it does not share a network association.

B.3.1.3 Effect on Receipt

On receipt of the INTRP-DATA.request primitive by the stub APS, the stub APS will construct and transmit a frame containing the given ASDU and other parameters using the MCPS-DATA.request primitive of the MAC sub-layer, as described in sub-clause B.5.1, and, once the corresponding MCPS-DATA.confirm primitive is received, Generate the INTRP-DATA.confirm primitive with a status value reflecting the status value returned by the MAC.

B.3.2 The INTRP-DATA.confirm Primitive

The INTRP-DATA.confirm primitive allows the stub APS to inform the application entity about the status of a data request.

B.3.2.1 Semantics of the Service Primitive

The primitive interface is as follows:

INTRP-DATA.confirm	{
	ASDUHandle
	Status
	}

Parameters of the primitive appear in Table B.2.

Table B.2 Parameters of the INTRP-DATA.confirm

Name	Type	Valid Range	Description
ASDUHandle	Integer	0x00 – 0xff	An integer handle associated with the transmitted frame.
Status	Enumeration	Any Status value returned by the MAC	The status of the ASDU transmission corresponding to ASDUHandle as returned by the MAC.

B.3.2.2 When Generated

This primitive is generated by the stub APS on a ZigBee device and passed to the application in response to the receipt of a MCPS-DATA.confirm primitive that is a confirmation of a previous MCPS-DATA.request issued by the stub APS.

B.3.2.3 Effect on Receipt

As a result of the receipt of this primitive, the application is informed of the results of an attempt to send a frame via the stub APS.

B.3.3 The INTRP-DATA.indication Primitive

The INTRP-DATA.indication primitive allows the stub APS to inform the next higher layer that it has received a frame that was transmitted via the stub APS on another device.

B.3.3.1 Semantics of the Service Primitive

The primitive interface is as follows:

INTRP-DATA.indication	{
	SrcAddrMode
	SrcPANId
	SrcAddress
	DstAddrMode
	DstPANId
	DstAddress
	ProfileId
	ClusterId
	ASDULength
	ASDU
	LinkQuality
	}

Parameters of the primitive appear in Table B.3.

Table B.3 Parameters of the INTRP-DATA.indication

Name	Type	Valid Range	Description
SrcAddrMode	Integer	0x03	The addressing mode for the source address used in this primitive. This parameter shall only reference the use of the 64-bit extended address: 0x03 = 64-bit extended address
SrcPANId	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the entity from which the ASDU is being transferred.
SrcAddress	64-bit address	As specified by the SrcAddrMode parameter	The device address of the entity from which the ASDU is being transferred.
DstAddrMode	Integer	0x01 – 0x03	The addressing mode for the destination address used in this primitive. This parameter can take one of the values from the following list: 0x01 = 16-bit group address 0x02 = 16-bit NWK address, normally the broadcast address 0xffff 0x03 = 64-bit extended address
DstPANID	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the entity or entities to which the ASDU is being transferred or the broadcast PAN ID 0xffff.
DstAddress	16-bit or 64-bit address	As specified by the DstAddrMode parameter	The address of the entity or entities to which the ASDU is being transferred.
ProfileId	Integer	0x0000 – 0xffff	The identifier of the application profile for which this frame is intended.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the cluster, within the profile specified by the ProfileId parameter, which defines the application semantics of the ASDU.

Table B.3 Parameters of the INTRP-DATA.indication (Continued)

Name	Type	Valid Range	Description
ASDULength	Integer	0x00 – (*aMaxMACFrameSize* - 9)	The number of octets in the ASDU to be transmitted.
ASDU	Set of octets	-	The set of octets forming the ASDU to be transmitted.
LinkQuality	Integer	0x00 – 0xff	The link quality observed during the reception of the ASDU.

B.3.3.2 When Generated

This primitive is generated and passed to the application in the event of the receipt, by the stub APS, of a MCPS-DATA.indication primitive from the MAC sub-layer, containing a frame that was generated by the stub APS of a peer ZigBee device, and that was intended for the receiving device.

B.3.3.3 Effect on Receipt

Upon receipt of this primitive the application is informed of the receipt of an application frame transmitted, via the stub APS, by a peer device and intended for the receiving device.

B.3.4 Qualifying and Testing of Inter-Pan Messages

Certification and application level testing shall ensure both the sending and receiving devices correctly react and understand the INTRP-DATA.request and INTRP-DATA.indication primitives.

B.4 Frame Formats

The birds-eye view of a normal ZigBee frame is as shown in Figure B.2.

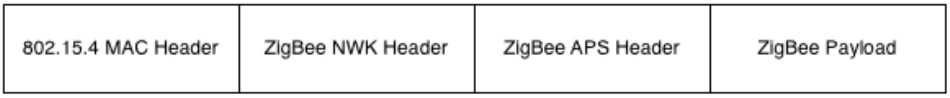

Figure B.2 Normal ZigBee Frame

Briefly, the frame contains the familiar headers controlling the operation of the MAC sub-layer, the NWK layer and the APS. Following these, there is a payload, formatted as specified in [B1].

Since most of the information contained in the NWK and APS headers is not relevant for inter-PAN transmission, the inter-PAN frame, shown Figure B.3, contains only a stub of the NWK header the APS header, which provide the information required by the stub APS shown in Figure B.4 to do its job.

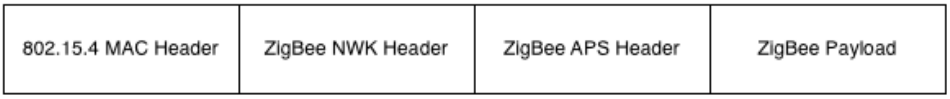

Figure B.3 Inter-PAN ZigBee Frame

Octets: 2
NWK frame control

Figure B.4 Stub NWK Header Format

The format of the frame control field of the stub NWK header is formatted as shown in Figure B.5.

Bits: 0-1	**2-5**	**6-15**
Frame type	Protocol version	Remaining sub-fields == 0

Figure B.5 Format of the NWK Frame Control Field

The sub-fields of the NWK frame control field are as follows:

- The frame type sub-field shall have a value of 0b11, which is a reserved frame type with respect to the [B3].
- The value protocol version sub-field shall reflect the protocol version of the ZigBee stack as described in [B3].

All other sub-fields shall have a value of 0.

The format of the stub APS header is shown in Figure B.6.

Octets: 1	0/2	2	2
APS frame control	Group address	Cluster identifier	Profile identifier
	Addressing fields		

Figure B.6 Stub APS Header Format

The stub APS header contains only 4 fields totaling a maximum of 7 octets in length.

The APS frame control field shall be 1 octet in length and is identical in format to the frame control field of the general APDU frame in [B3] (see Figure B.7).

Bits: 0-1	2-3	4	5	6	7
Frame type	Delivery Mode	Reserved	Security	ACK request	Extended Header Present

Figure B.7 Format of the APS Frame Control Field

The fields of the frame control field have the following values:

- The frame type sub-field shall have a value of 0b11, which is a reserved frame type with respect to the [B3].
- The delivery mode sub-field may have a value of 0b00, indicating unicast, 0b10, indicating broadcast or 0b11 indicating group addressing.
- Security is never enabled for Inter-Pan transmissions. This sub-field shall be a value of 0.
- The ACK request sub-field shall have a value of 0, indicating no ACK request. No APS ACKs are to be used with Inter-Pan transmissions.
- The extended header present sub-field shall always have a value of 0, indicating no extended header.

The optional group address shall be present if and only if the delivery mode field has a value of 0x0b11. If present it shall contain the 16-bit identifier of the group to which the frame is addressed.

The cluster identifier field is 2 octets in length and specifies the identifier of the cluster to which the frame relates and which shall be made available for filtering and interpretation of messages at each device that takes delivery of the frame.

The profile identifier is two octets in length and specifies the ZigBee profile identifier for which the frame is intended and shall be used during the filtering of messages at each device that takes delivery of the frame.

B.5 Frame Processing

Assuming the INTRP-SAP described above, frames transmitted using the stub APS are processed as described here.

B.5.1 Inter-PAN Transmission

On receipt of the INTRP-DATA.request primitive, the stub APS shall construct a stub APS frame. The header of the stub APS frame shall contain a NWK and an APS frame control field as described in clause B.4, a cluster identifier field equal to the value of the ClusterId parameter of the INTRP-DATA.request and a profile identifier field equal to the value of the ProfileId parameter. If the DstAddrMode parameter of the INTRP-DATA.request has a value of 0x01, indicating group addressing, then the APS header shall also contain a group address field with a value corresponding to the value of the DstAddress parameter. The payload of the stub APS frame shall contain the data payload to be transmitted.

The stub APS frame will then be transmitted using the MCPS-DATA.request primitive of the MAC sub-layer with key primitive parameters set as follows:

- The value of the SrcAddrMode parameter of the MCPS-DATA.request shall always be set to a value of three, indicating the use of the 64-bit extended address.
- The SrcPANId parameter shall be equal to the value of the *macPANID* attribute of the MAC PIB.
- The SrcAddr parameter shall always be equal to the value of the MAC sub-layer constant *aExtendedAddress*.
- If the DstAddrMode parameter of the INTRP-DATA.request primitive has a value of 0x01, then the DstADdrMode parameter of the MCPS-DATA.request shall have a value of 0x02. Otherwise, the DstAddrMode parameter of the MCPS-DATA.request shall reflect the value of the DstAddrMode parameter of the INTRP-DATA.request.
- The DstPANId parameter shall have the value given by the DstPANID parameter of the INTRP-DATA.request primitive.
- If the DstAddrMode parameter of the INTRP-DATA.request has a value of 0x01, indicating group addressing, then the value of the DstAddr parameter of the MCPS-DATA.request shall be the broadcast address 0xffff. Otherwise, value of the DstAddr parameter shall reflect the value of the DstAddress parameter of the INTRP-DATA.request primitive.
- The MsduLength parameter shall be the length, in octets, of the stub APS frame.

- The Msdu parameter shall be the stub APS frame itself.
- If the transmission is a unicast, then the value of the TxOptions parameter shall be 0x01, indicating a request for acknowledgement. Otherwise, the TxOptions parameter shall have a value of 0x00, indicating no options.

On receipt of the MCPS-DATA.confirm primitive form the MAC sub-layer, the stub APS will invoke the transmit confirmation function with a status reflecting the status returned by the MAC.

B.5.2 Inter-PAN Reception

On receipt of the MCPS-DATA.indication primitive from the MAC sub-layer, the receiving entity - in case of a ZigBee device this is normally the NWK layer - shall determine whether the frame should be passed to the stub APS or processed as specified in [B3]. For a frame that is to be processed by the stub APS, the non-varying sub-fields of both the NWK frame control field and the APS frame control field must be set exactly as described above.

If the delivery mode sub-field of the APS frame control field of the stub APS header has a value of 0b11, indicating group addressing, then, if the device implements group addressing, the value of the group address field shall be checked against the NWK layer group table, and, if the received value is not present in the table, the frame shall be discarded with no further processing or action.

On receipt of a frame for processing, the stub APS shall generate an INTRP-DATA.indication with parameter values as follows:

- The value of the SrcAddrMode parameter of the INTRP-DATA.indication shall always be set to a value of three, indicating the use of the 64-bit extended address
- The value of the SrcPANId parameter shall reflect that of the SrcPANId parameter of the MCPS-DATA.indication.
- The SrcAddress parameter of the INTRP-DATA.indication shall always reflect the value of a 64-bit extended address.
- Values for the DstAddrMode parameter shall be one of:
  - 0x03, if the DstAddrMode parameter of the INTRP-DATA.indication has a value of 0x03.
  - 0x02, if the DstAddrMode parameter of the INTRP-DATA.indication has a value of 0x02
- The value of the DstPANId parameter of the INTRP-DATA.indication shall reflect the value of the DstPANId parameter of the MCPS-DATA.indication.

- If the DstAddrMode parameter of the INTRP-DATA.indication has a value of 0x01, indicating group addressing then the DstAddress parameter of the INTRP-DATA.indication shall reflect the value of the group address field of the stub APS header. Otherwise, the value of the DstAddress parameter of the INTRP-DATA.indication shall reflect the value of the DstAddr parameter of the MCPS-DATA.indication.
- The value of the ProfileId parameter shall be the same as the value of the profile identifier field of the stub APS header.
- The value of the ClusterId parameter shall be the same as the value of the cluster identifier field of the stub APS header.
- The ASDULength field shall contain the number of octets in the stub APS frame payload.
- The ASDU shall be the stub APS payload itself.
- The value of the LinkQuality parameter shall reflect the value of the mpduLinkQuality parameter of the MCPS-DATA.indication.

B.6 Usage Scenario

Figure B.8 shows a typical usage scenario for inter-PAN communication. In this Smart Energy-oriented scenario, the Home Area Network (HAN) device is on the left with the APL, NWK and MAC shown as separate sequences. A ZigBee electric meter or Energy Service Interface (ESI)[129] is also shown along with a “foreign”, i.e. non-ZigBee, device.

129.CCB 1072

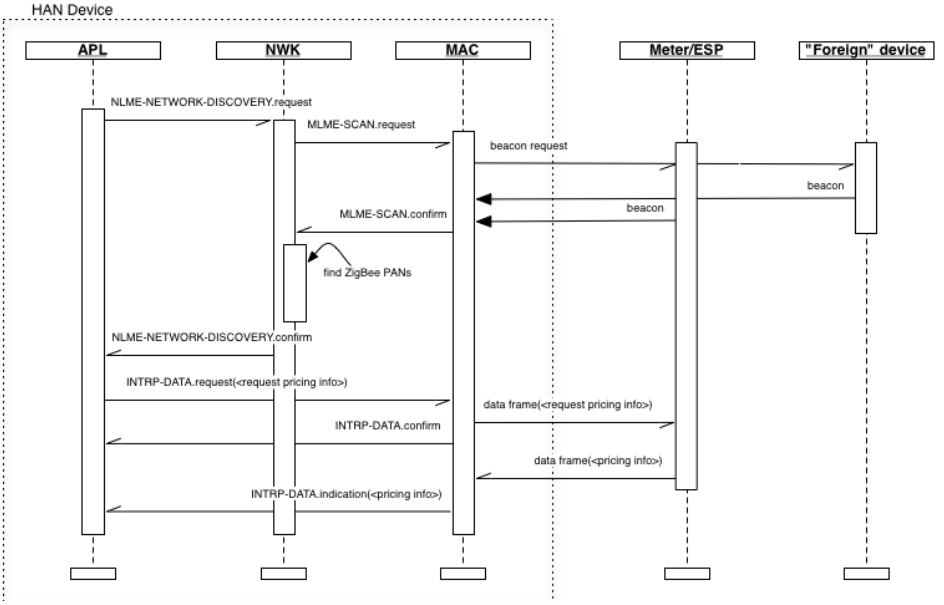

Figure B.8 Inter-PAN Typical Usage

The first task of the HAN device in this scenario is to discover devices in the area that are capable of publishing pricing information. It could do this using an inter-PAN broadcast, i.e. a broadcast employing both the broadcast address and the broadcast PAN ID, but in doing this it runs the risk of confusing the non-ZigBee "foreign" device. As an alternative, the HAN device uses standard ZigBee network discovery (see [B3]) in order to find ZigBee PANs.

Once at least one ZigBee PAN is discovered, the HAN device sends a request for public pricing information using the INTRP-DATA SAP. Typically, the first time this request is sent, it will be sent as a broadcast to each discovered ZigBee PAN. Receiving devices that implement the INTRP-DATA SAP will process it and, if any such device is able to respond, it will respond directly to the requestor. After receiving at least one response the requestor may store the PAN ID and device address of one or more responders so that it may query them directly in the future.

B.7 Best Practices[130]

Network Channel Manager Inter-PAN support is not specified in Annex E of the core stack specification ([B3]). New channel notifications will not be broadcast

130.CCB 923

Inter-PAN. Inter-PAN devices which do not receive the network channel change will need to perform the network discovery procedure described in B.3.4.[131]

B.8 Security Requirements[132]

SE devices supporting Inter-PAN shall not allow access to any other cluster except those specifically described below. All other Smart Energy messages received over Inter-PAN shall be dropped.

Frames allowed to be received over Inter-PAN

1 Get Current Price

2 Get Scheduled Prices

3 Get Last Message

Frames allowed to be sent over Inter-PAN

1 Publish Price

2 Display Message

3 Cancel Message

In addition, devices shall verify the correct format of all SE messages received over Inter-PAN. Any received message that does not conform to the format described in this document shall be dropped.

131.CCB 994

132.CCB 994

This page intentionally blank

ANNEX

C

KEY ESTABLISHMENT CLUSTER

The candidate material in this annex, when approved, will be merged into the Foundation document of the ZigBee Cluster Library (ZCL) by the Cluster Library Development Board.

C.1 Scope and Purpose

This Annex specifies a cluster, which contains commands and attributes necessary for managing secure communication between ZigBee devices.

This Annex should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see [B1]), 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 [B4] will continue to be developed in a backward-compatible manner as a more general secure communication cluster for ZigBee applications as a whole.

C.2 General Description

C.2.1 Introduction

As previously stated, this document describes a cluster for managing secure communication in ZigBee. The cluster is for Key Establishment.

C.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.

C.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 [B15] 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 either be effected 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 which can be implemented:

- Symmetric Key Key Establishment
- Public Key Key Establishment

C.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.

C.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 C.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.

C.2.6 General Exchange

The following diagram shows an overview of the general exchange which takes place between initiator and responder to perform key establishment.

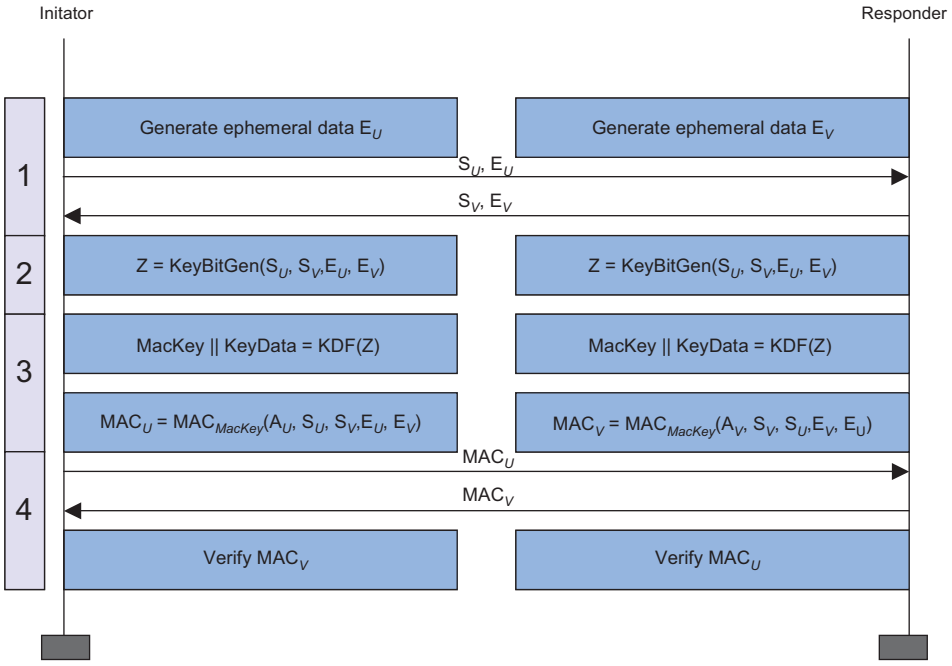

Figure C.1 Overview of General Exchange

The functions are as follows:

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 the diagram (Figure C.1)[133] depend on the Key Establishment mechanism.

C.2.6.1 Exchange Static and Ephemeral Data

Figure C.1 shows static data S_U and S_V. For PKKE schemes, this represents a combination of the 64-bit device address [B11] 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 C.1 also shows ephemeral data E_U and E_V. For PKKE schemes, this represents the public key of a randomly generated key pair.

133.CCB 1159

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*.

C.2.6.2 Generate Key Bitstream

Figure C.1 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 [B18]. 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*.

C.2.6.3 Derive MAC Key and Key Data

Figure C.1 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 [B18]. Note there is no *SharedInfo* parameter of the referenced KDF, i.e. it is a null octet string of length 0.

Figure C.1 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 [B18]. The MAC in the reference is the keyed hash function for message authentication specified in sub-clause C.4.2.2.6 and the message *M* is a concatenation of the identity (the 64-bit device address [B11]) 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.

C.2.6.4 Confirm Key Using MAC

Figure C.1 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.

C.3 Cluster List

The clusters specified in this document are listed in Table C.1.

For our purposes, any device that implements the client side of this cluster may be considered the initiator of the secure communication transaction.

Table C.1 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.

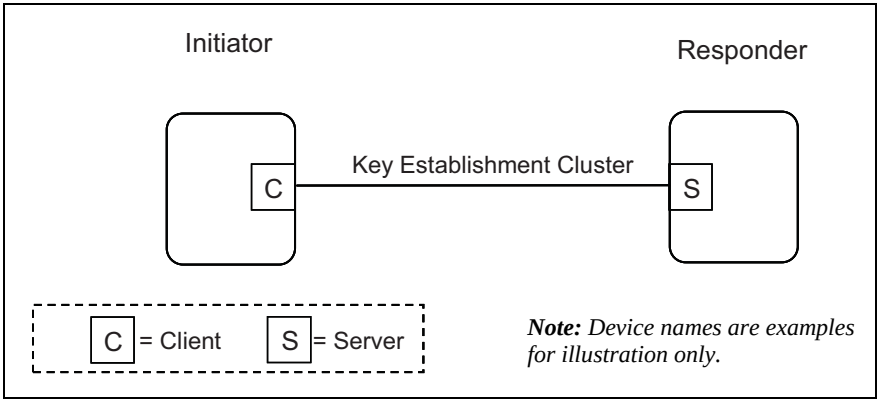

Figure C.2 Typical Usage of the Key Establishment Cluster

C.3.1 Key Establishment Cluster

C.3.1.1 Overview

This cluster provides attributes and commands to perform mutual authentication and establish keys between two ZigBee devices. Figure C.3 depicts a diagram of a successful key establishment negotiation.

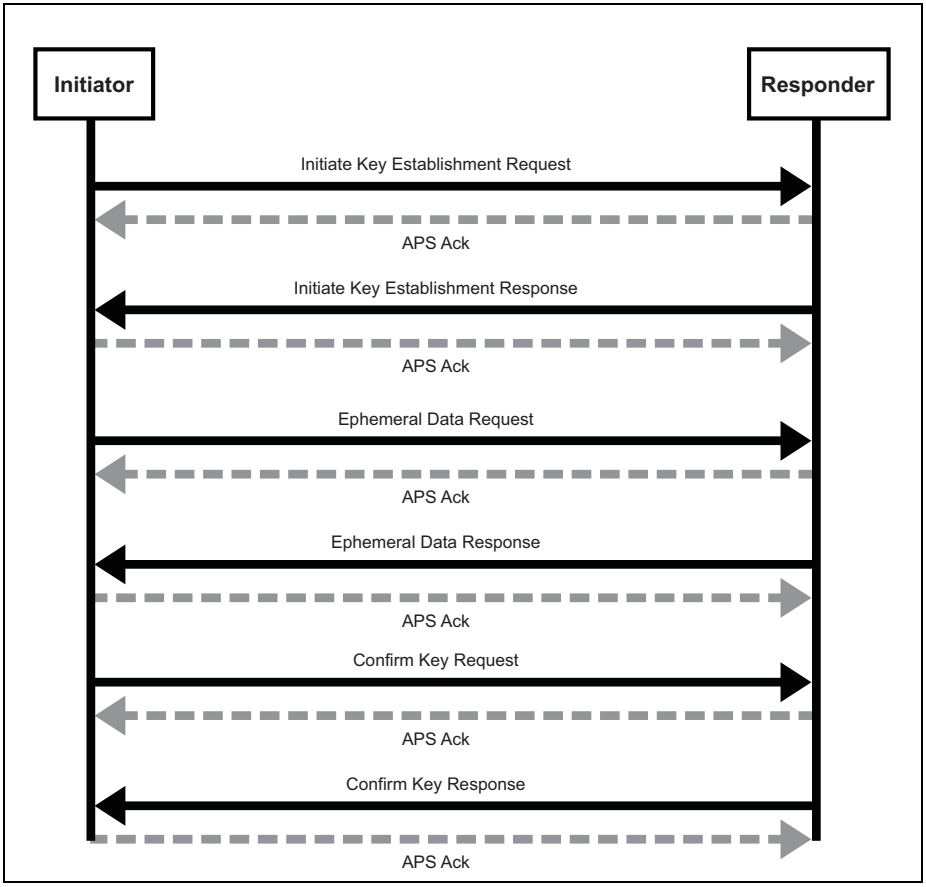

Figure C.3 Key Establishment Command Exchange

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.

C.3.1.2 Server

C.3.1.2.1 Dependencies

The Key Establishment server cluster has no dependencies.

C.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 C.2.

Table C.2 Key Establishment Attribute Sets

Attribute Set Identifier	Description
0x000	Information
0x001 – 0xfff	Reserved

C.3.1.2.2.1 Information

The Information attribute set contains the attributes summarized in Table C.3.

Table C.3 Key Establishment Attribute Sets

Identifier	Name	Type	Range	Access	Default	Mandatory/ Optional
0x0000	*KeyEstablishmentSuite*	16-bit Enumeration	0x0000 - 0xFFFF	Read only	0x0000	M

C.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 C.4 Values of the *KeyEstablishmentSuite* Attribute

Bits	Description
0	Certificate-based Key Establishment (CBKE-ECMQV)
1-15	Reserved

C.3.1.2.3 Commands Received

The server side of the key establishment cluster is capable of receiving the commands listed in Table C.5.

Table C.5 Received Command IDs for the Key Establishment Cluster Server

Command Identifier Field Value	Description	Mandatory/ Optional
0x00	*Initiate Key Establishment Request*	M
0x01	*Ephemeral Data Request*	M
0x02	*Confirm Key Data Request*	M
0x03	*Terminate Key Establishment*	M
0x04 – 0xFF	Reserved	

C.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.

C.3.1.2.3.1.1 Payload Format

The *Initiate Key Establishment Request* command payload shall be formatted as illustrated in Figure C.4.

Octets	**2**	**1**	**1**	**48**
Data Type	16-bit BitMap	Unsigned 8-bit Integer	Unsigned 8-bit Integer	Octets (non-ZCL Data Type)
Field Name	Key Establishment suite	Ephemeral Data Generate Time	Confirm Key Generate Time	Identity (IDU)

Figure C.4 Format of the *Initiate Key Establishment Request* Command Payload

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[134]**:** 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[135]**:** 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 C.4.2.

C.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

134. CCB 993
135. CCB 993

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[136] 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.[137]

C.3.1.2.3.2 Ephemeral Data Request Command[138]

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.

C.3.1.2.3.2.1 Payload Format

Octets	**22**
Data Type	Octets (non-ZCL Data Type)
Field Name	Ephemeral Data (QEU)

Figure C.5 Format of the *Ephemeral Data Request* Command Payload

C.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.[139]

136. CCB 1125
137. CCB 1159
138. CCB 1159
139. CCB 1159

C.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.

C.3.1.2.3.3.1 Payload Format

The *Confirm KeyRequest* command payload shall be formatted as illustrated in Figure C.6.

Octets	**16**
Data Type	Octets (non-ZCL Data Type)[a]
Field Name	Secure Message Authentication Code (*MACU*)

a. CCB 1098

Figure C.6 Format of the *Confirm Key Request* Command Payload

Secure Message Authentication Code field: The Secure Message Authentication Code field shall be the octet[140] representation of *MACU* as specified in sub-clause C.4.2.

C.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.

140.CCB 1098

C.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.

C.3.1.2.3.4.1 Payload Format

The *Terminate Key Establishment* command payload shall be formatted as illustrated in Figure C.7.

Octets	**1**	**1**	**2**
Data Type	8-bit Enumeration	Unsigned 8-bit Integer	16-bit BitMap
Field Name	Status Code	Wait Time	KeyEstablishmentSuite

Figure C.7 Format of the *Terminate Key Establishment* Command Payload

Status Field: The Status field shall be one of the error codes in Table C.6.

Table C.6 *Terminate Key Establishment* Command Status Field

Enumeration	Value	Description
	0x00	Reserved
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.
	0x06 - 0xFF	Reserved

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.

C.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.[141]

C.3.1.2.4 Commands Generated

The server generates the commands detailed in sub-clause C.3.1.3.3, as well as those used for reading and writing attributes.

C.3.1.3 Client

C.3.1.3.1 Dependencies

The Key Establishment client cluster has no dependencies.

C.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 C.7.

141.CCB 1159

Table C.7 Key Establishment Attribute Sets

Attribute Set Identifier	Description
0x000	Information
0x001 – 0xfff	Reserved

C.3.1.3.2.1 Information

The Information attribute set contains the attributes summarized in Table C.8.

Table C.8 Attributes of the Information Attribute Set

Identifier	Name	Type	Range	Access	Default	Mandatory/ Optional
0x0000	*KeyEstablishmentSuite*	16-bit Enumeration	0x0000 – 0xFFFF	Read only	0x0000	M

C.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 C.9.

Table C.9 Values of the *KeyEstablishmentSuite* Attribute

KeyEstablishmentSuite	Description
0	Certificate-based Key Establishment (CBKE - ECMQV)
1-15	Reserved

C.3.1.3.3 Commands Received

The client side of the Key Establishment cluster is capable of receiving the commands listed in Table C.10.

Table C.10 Received Command IDs for the Key Establishment Cluster Client

Command Identifier Field Value	Description	Mandatory / Optional
0x00	*Initiate Key Establishment Response*	M
0x01	*Ephemeral Data Response*	M
0x02	*Confirm Key Data Response*	M
0x03	*Terminate Key Establishment*	M
0x04 - 0xFF	Reserved	

C.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.

C.3.1.3.3.1.1 Payload Format

The *Initiate Key Establishment Response* command payload shall be formatted as illustrated in Figure C.8.

Octets	2	1	1	48
Data Type	16-bit BitMap	Unsigned 8-bit Integer	Unsigned 8-bit Integer	Octets (non-ZCL Data Type)
Field Name	Requested Key Establishment suite	Ephemeral Data Generate Time	Confirm Key Generate Time	Identity (IDU)

Figure C.8 Format of the *Initiate Key Establishment Response* Command Payload

[142]**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.

142.CCB 993

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 C.4.2.

C.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[143] 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.[144]

C.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.

143.CCB 1125
144.CCB 1159

C.3.1.3.3.2.1 Payload Format

Octets	22
Data Type	Octets (non-ZCL Data Type)
Field Name	Ephemeral Data (QEV)

Figure C.9 Format of the *Ephemeral Data Response* Command Payload

C.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.

C.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.

C.3.1.3.3.3.1 Payload Format

The *Confirm Key Response* command payload shall be formatted as illustrated in Figure C.10.

Octets	16[a]
Data Type	Octets (non-ZCL Data Type)
Field Name	Secure Message Authentication Code (*MACV*)

a. CCB 1098

Figure C.10 Format of the *Confirm Key Response* Command Payload

Secure Message Authentication Code field: The Secure Message Authentication Code field shall be the octet[145] representation of *MACV* as specified in sub-clause C.4.2.

C.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 an *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.

C.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.

C.3.1.3.3.4.1 Payload Format

Octets	**1**	**1**	**2**
Data Type	8-bit Enumeration	Unsigned 8-bit Integer	16-bit BitMap
Field Name	Status Code	Wait Time	KeyEstablishmentSuite

Figure C.11 Format of the *Terminate Key Establishment* Command Payload

Status field: The Status field shall be one of the following error codes.

145.CCB 1098

Table C.11 *Terminate Key Establishment* Command Status Field

Enumeration	Value	Description
	0x00	Reserved
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.
	0x06 - 0xFF	Reserved

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.

C.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.

C.3.1.3.4 Commands Generated

The client generates the commands detailed in sub-clause C.3.1.2.3, as well as those used for reading and writing attributes.

C.4 Application Implementation

C.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 [B11].

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)[146] at the premises where the joining device is by installing the temporary link key at the ESI[147] 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[148] 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.

146.CCB 1072
147.CCB 1072
148.CCB 1072

C.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[149] key gets compromised none of future and past communications are exposed
- Known Key Security resilience: Each shared link key created per session is unique

149.CCB 1159

C.4.2.1 Notation and Representation

C.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.

C.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 [B18].

C.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 [B11]. The parameter *entlen* shall have the integer value 64.

C.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.

C.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 [B21].

All elliptic-curve points (and operations hereon) used in this specification shall be (performed) on this curve.

C.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 [B18]. Thus, each elliptic-curve point can be represented in 22 bytes.

C.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 [B18]. 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 [B18]. Each

public key shall be represented as defined in sub-clause C.4.2.1.2 of this document.

C.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 C.4.2.2.2 (see SEC4 [B18]);
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 [B11] 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 [B22]) shall be the concatenation of the *Subject*, *Issuer*, and *ProfileAttributeData*:

   I_U = *Subject* || *Issuer* || *ProfileAttributeData*

C.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 [B16]. This block-cipher has a key size that is equal to the block size, in bits, i.e., *keylen*= 128.

C.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 [B3], with the following instantiations:

1. Each entity shall use the block-cipher E as specified in sub-clause B.1.1 in [B3].
2. All integers and octets shall be represented as specified in sub-clause C.4.2.1.

The Matyas-Meyer-Oseas hash function (specified in Annex B.6 in [B3]) has a message digest size *hashlen* that is equal to the block size, in bits, of the established blockcipher.

C.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 [B17] with the following instantiations:

1. Each entity shall use the cryptographic hash *H* function as specified in sub-clause C.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 C.4.2.2.6.

C.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.

C.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 [B18], with the following instantiations:

1. Each entity shall be a DEV;
2. Each entity's identifier shall be its 64-bit device address [B11]; the parameter *entlen* shall have the integer value 64;
3. Each entity shall use the cryptographic hash function as specified in sub-clause C.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 [B18]);

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 [B18]).

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 C.12.

Table C.12 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 [B18]) 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 [B18]) 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 [B18]).
QEV	22	The ephemeral public key generated by the responder device (denoted Q_{2V} in the Elliptic Curve MQV scheme in SEC1 [B18]).
MACU	16	The secure message authentication code generated by the initiator device (where the message M is $(02_{16} \| ID_U \| ID_V \| QEU \| QEV)$ 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 C.4.2.2.2. See also section 3.7 of SEC1 [B18]).
MACV	16	The secure message authentication code generated by the responder device (where the message M is $(03_{16} \| ID_V \| ID_U \| QEV \| QEU)$ 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 C.4.2.2.3. See also section 3.7 of SEC1 [B18]).

C.4.2.3.1 Exchange Ephemeral Data

C.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.

C.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.

C.4.2.3.2 Validate Implicit Certificates

C.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 C.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 [B21].

C.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 C.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 [B21].

C.4.2.3.3 Derive Keying Material

C.4.2.3.3.1 Initiator

The initiator performs the Elliptic Curve MQV scheme as specified in section 6.2 of SEC1 [B18] with the following instantiations:

1. The elliptic curve domain parameters shall be as specified in sub-clause C.4.2.2.1;
2. The KDF shall use the cryptographic hash function specified in sub-clause C.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 [B18] 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.

C.4.2.3.3.2 Responder

The responder performs the Elliptic Curve MQV scheme as specified in section 6.2 of SEC1 [B18] with the following instantiations:

1 The elliptic curve domain parameters shall be as specified in sub-clause C.4.2.2.1;

2 The KDF shall use the cryptographic hash function specified in sub-clause C.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 [B18] 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.

C.4.2.3.4 Confirm Keys

C.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.

C.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.

C.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).

C.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.

C.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
```

C.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
```

C.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 4B C1 B4 22 39 83 07
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

C.5.2 Key Establishment Messages

The following is the basic flow of messages back and forth between the initiator and the responder performing key establishment using the Key Establishment Cluster.

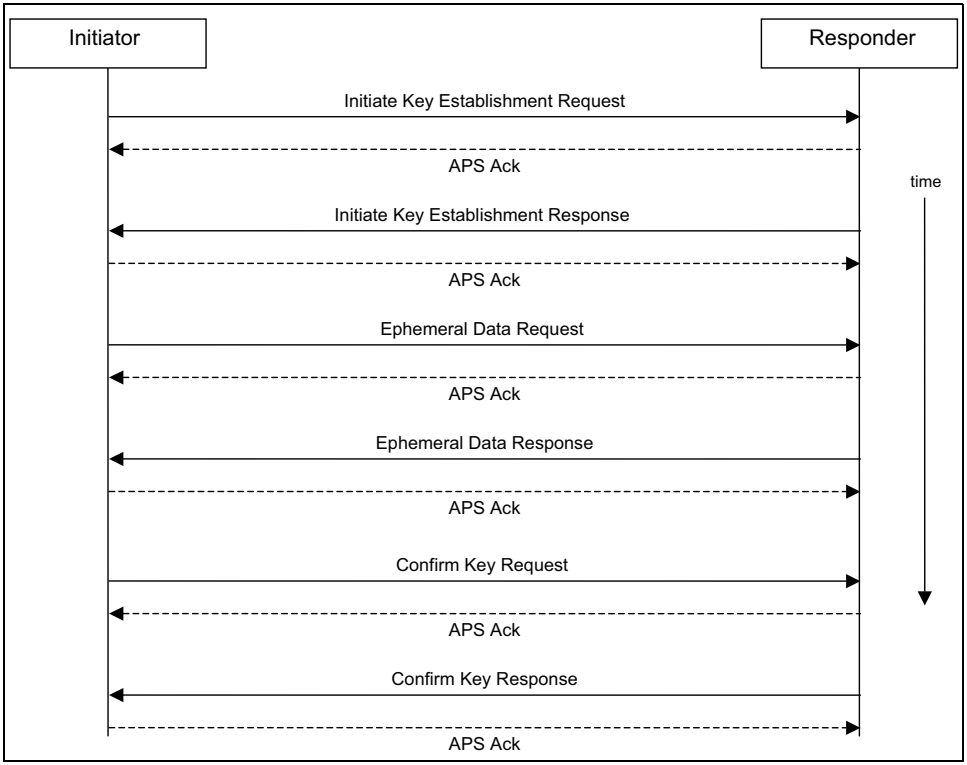

Figure C.12 Key Establishment Command Exchange

C.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

C.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

C.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	

C.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	

C.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	

C.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`	

C.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*

C.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.

C.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.

C.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).

C.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

C.5.3.3.2 Step Summary

1. Derive the Shared Secret using the ECMQV primitives
   a. Z = ECC_GenerateSharedSecret(Private(U), E(U), E-P(U), Cert(V), E(V), CA)
2. Derive the Keying data

a. Hash-1 = Z || 00 00 00 01 || SharedData

b. Hash-2 = Z || 00 00 00 02 || SharedData

3. Parse KeyingData as follows

a. MacKey = First 128 bits (Hash-1) of KeyingData

b. KeyData = Second 128 bits (Hash-2) of KeyingData

4. Create MAC(U)

a. 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)'

a. MAC(V) = MAC(MacKey) { M(V) || ID(V) || ID(U) || E(V) || E(U) }

8. Verify MAC(V)' is the same as MAC(V).

C.5.3.3.3 Detailed Steps

1. Derive the Shared Secret using the ECMQV primitives

a. 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

a. 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
```

b. 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

a. MacKey = First 128 bits (Hash-1) of KeyingData

b. KeyData = Second 128 bits (Hash-2) of KeyingData

4. Create MAC(U)

a. 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)'

a. 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).

C.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).

C.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

C.5.3.4.2 Step Summary

1. Derive the Shared Secret using the ECMQV primitives
   a. Z = ECC_GenerateSharedSecret(Private(V), E(V), E-P(V), Cert(U), E(U), CA)
2. Derive the Keying data
   a. Hash-1 = Z || 00 00 00 01 || SharedData
   b. Hash-2 = Z || 00 00 00 02 || SharedData
3. Parse KeyingData as follows
   a. MacKey = First 128 bits (Hash-1) of KeyingData
   b. KeyData = Second 128 bits (Hash-2) of KeyingData
4. Create MAC(V)
   a. MAC(V) = MAC(MacKey) { M(V) || ID(V) || ID(U) || E(V) || E(U) }
5. Calculate MAC(U)'
   a. 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.

C.5.3.4.3 Detailed Steps

1. Derive the Shared Secret using the ECMQV primitives
   a. 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
   a. 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
```

   b. 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

a MacKey = First 128 bits (Hash-1) of KeyingData

b KeyData = Second 128 bits (Hash-2) of KeyingData

4 Create MAC(V)

a 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
```

5 Calculate MAC(V)'

a 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
```

6 Verify MAC(V)' is the same as MAC(V).

7 Send MAC(V) to U.[150]

150.CCB 984

This page intentionally blank

ANNEX

D

SMART ENERGY CLUSTER DESCRIPTIONS

The candidate material in this annex describing the Smart Energy Clusters, when approved, will be merged into the Foundation document of the ZigBee Cluster Library (ZCL) by the Cluster Library Development Board.

D.1 Annex Guidelines

D.1.1 Client/Server Model Information

The ZigBee Cluster Library Specification is used as the guiding reference for defining the rule set in defining the Client/Server model for the Smart Energy Profile. Please note the following items influence the further refinement of that definition:

- Attributes can be defined for both Client and Server side clusters. Attributes can be used to understand current state of activities within a device, enhancing both the diagnostic and maintenance of devices or the processes supported by that device.
- The ESI[151] device acts as the transition point from upstream Wide Area Network (and subsequent upstream systems) to the ZigBee network. Because of this responsibility, in some of the clusters it acts as a proxy for the upstream systems. In situations in which the proxy condition occurs, plus where attributes are defined or commands (transactions) are initiated on both client/server sides, the ESI[152] will be by default labeled as the Server side in the cluster descriptions.

151.CCB 1072
152.CCB 1072

D.1.2 Interpretation of Reserved Field Values or Bitmaps

To support backwards compatibility, devices should ignore any values or bit settings for any reserved field values. If the field is necessary for interpretation or in conjunction with other fields[153] the whole message can be ignored.

To enable future growth and ensure backwards compatibility, any existing devices which encounter any fields applied after the end of a command shall treat them as reserved fields. The future addition of fields applied after the end of defined cluster commands are reserved solely for ZigBee specifications, Manufacturers shall not add fields after the end of commands.[154]

D.2 Demand Response and Load Control Cluster

D.2.1 Overview

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.

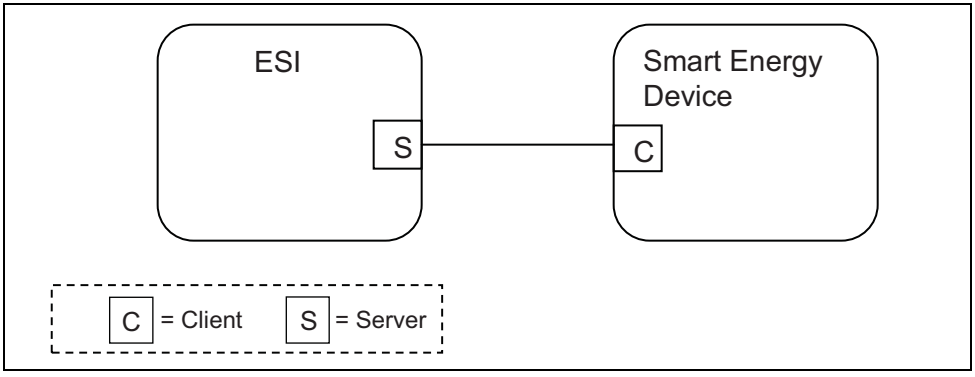

Figure D.1 Demand Response/Load Control Cluster Client Server Example

Please note the ESI[155] 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.

153.CCB 1294
154.CCB 968
155.CCB 1072

D.2.2 Server

By default the ESI[156] will be labeled as the Server side in the cluster descriptions, being able to initiate load control commands to other devices in the network.

D.2.2.1 Dependencies

A server device shall be capable of storing at least two load control events. [157]

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[158] as described in sub-clause 5.12.1.1.

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.

D.2.2.2 Attributes

There are no attributes for the Demand Response and Load Control Cluster server.

D.2.2.3 Commands Generated

The command IDs generated by the Demand Response and Load Control cluster server are listed in Table D.1.

Table D.1 Command IDs for the Demand Response and Load Control Server

Command Identifier Field Value	Description	Mandatory/ Optional
0x00	*Load Control Event*	M
0x01	*Cancel Load Control Event*	M
0x02	*Cancel All Load Control Events*	M
0x03 – 0xff	Reserved	

156.CCB 1072
157.CCB 1318
158.CCB 1072

D.2.2.3.1 *Load Control Event* Command

D.2.2.3.1.1 Payload Format

The *Load Control Event* command payload shall be formatted as illustrated in Figure D.2.

Octets	4	2	1	4	2	1	1
Data Type	Unsigned 32-bit integer	16-bit BitMap	Unsigned 8-bit integer	UTC Time	Unsigned 16-bit integer	Unsigned 8-bit integer	Unsigned 8-bit integer
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	Unsigned 8-bit integer	Signed 16-bit integer	Signed 16-bit integer	Signed 8-bit integer	Unsigned 8-bit integer	8-bit BitMap
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)

Figure D.2 Format of the *Load Control Event* Command Payload

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.*

D.2.2.3.1.1.1 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 (UTCTime 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 D.2:

Table D.2 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
12 to 15	Reserved

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 D.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 *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.[159]

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.[160]

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 D.3.

Table D.3 Criticality Levels

Criticality Level	Level Description	Participation
0	Reserved	
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
0x10 to 0xFF	Reserved	

159. CCB 1380
160. CCB 1243

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 [B1].

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[161], 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.

161.CCB 1346

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.[162]

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 minutes7 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 D.4.

Table D.4 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
2 to 7	Reserved

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 D.2.3.2.2 and sub-clause D.2.3.2.3 for the settings of these values.*

D.2.2.3.1.1.2 When Generated

This command is generated when the ESI[163] wants to control one or more load control devices, usually as the result of an energy curtailment command from the Smart Energy network.

162.CCB 1441
163.CCB 1072

D.2.2.3.1.1.3 Responses to Load Control Event

The server receives the cluster-specific commands detailed in sub-clause D.2.3.3.1.

D.2.2.3.2 *Cancel Load Control Event* Command

D.2.2.3.2.1 Payload Format

The *Cancel Load Control Event* command payload shall be formatted as illustrated in Figure D.3.

Octets	4	2	1	1	4
Data Type	Unsigned 32-bit integer	16-bit BitMap	Unsigned 8-bit integer	8-bit BitMap	UTCTime
Field Name	Issuer Event ID	Device Class (M)	Utility Enrollment[a] Group (M)	Cancel Control (M)[b]	Effective Time (M)[c]

a. CCB 1322
b. CCB 1322
c. CCB 1322

Figure D.3 Format of the *Cancel Load Control Event* Payload

D.2.2.3.2.1.1 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 (UTCTime 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 D.2.

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 D.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.[164]

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.[165]

Cancel Control (mandatory): The encoding of the Cancel Control is listed in Table D.5.

Table D.5 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.
1 to 7	Reserved

Effective Time (mandatory): UTC Timestamp representing when the canceling of the event is scheduled to start. An effective[166] 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.[167]

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.* [168]

164. CCB 1294
165. CCB 1456
166. CCB 1294
167. CCB 1243
168. CCB 1455

D.2.2.3.2.1.2 When Generated

This command is generated when the ESI[169] 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.

D.2.2.3.2.1.3 Responses to *Cancel Load Control Event*

The server receives the cluster-specific commands detailed in sub-clause D.2.3.3.1.

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)".*[170]

D.2.2.3.3 *Cancel All Load Control Events* Command

D.2.2.3.3.1 Payload Format

The *Cancel All Load Control Events* command payload shall be formatted as illustrated in Figure D.4.

Octets	**1**
Data Type	8-bit BitMap
Field Name	Cancel Control

Figure D.4 Format of the *Cancel All Load Control Events* Command Payload

D.2.2.3.3.1.1 Payload Details

Cancel Control: The encoding of the Cancel Control is listed in Table D.6.

169.CCB 1072
170.CCB 1348

Table D.6 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.
1 to 7	Reserved

D.2.2.3.3.2 When Generated

This command is generated when the ESI[171] wants to cancel all events for control device(s).

D.2.2.3.3.3 Responses to *Cancel All Load Control Events*

The server receives the cluster-specific commands detailed in sub-clause D.2.3.3.1. 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.

D.2.2.4 Commands Received

The server receives the cluster-specific commands detailed in sub-clause D.2.3.3.[172]

D.2.3 Client

This section identifies the attributes and commands provided by Client devices.

D.2.3.1 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

171.CCB 1072
172.CCB 1028

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[173] as described in sub-clause 5.12.1.1.

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).[174]

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.

D.2.3.2 Client Cluster Attributes

Table D.7 Demand Response Client Cluster Attributes

Identifier	Name	Type	Range	Access	Default	Mandatory/ Optional
0x0000	*UtilityEnrollment Group*	Unsigned 8-bit Integer	0x00 to 0xFF	Read/ Write	0x00	M
0x0001	*StartRandomizeMinutes*	Unsigned 8-bit Integer	0x00 to 0x3C	Read/ Write	0x1E	M
0x0002	*StopRandomizeMinutes*	Unsigned 8-bit Integer	0x00 to 0x3C	Read/ Write	0x1E	M
0x0003	*DeviceClassValue*	Unsigned 16-bit Integer	0x0000 to 0xFFFF[a]	Read/ Write[b]	-	M
0x0004 to 0xFFFF	Reserved					

a. CCB 1316

b. CCB 1135

173.CCB 1072
174.CCB 1449

D.2.3.2.1 *Utility Enrollment Group* 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.

D.2.3.2.2 *Start Randomization Minutes* 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.

D.2.3.2.3 *End Randomization Minutes* 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.

D.2.3.2.4 *DeviceClassValue* Attribute

The *DeviceClassValue* attribute identifies which bits the device will match in the Device Class fields. Please refer to Table D.2, “Device Class Field BitMap/ Encoding” for further details. Although the attribute has a read/write 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.[175]

Although, for backwards compatibility, the Type cannot be changed, this 16-bit Integer should be treated as if it were a 16-bit BitMap.[176]

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

175.CCB 1437
176.CCB 1316

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.[177]

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.[178]

D.2.3.3 Commands Generated

The command IDs generated by the Demand Response and Load Control client cluster are listed in Table D.8.

Table D.8 Generated Command IDs for the Demand Response and Load Control Client

Command Identifier Field Value	Description	Mandatory/ Optional
0x00	*Report Event Status*	M
0x01	*Get Scheduled Events*	M
0x02 – 0xff	Reserved	

D.2.3.3.1 *Report Event Status* Command

D.2.3.3.1.1 Payload Format

The *Report Event Status* command payload shall be formatted as illustrated in Figure D.5.

177.CCB 1456
178.CCB 1457

Octets	4	1	4	1	2	2
Data Type	Unsigned 32-bit integer	Unsigned 8-bit integer	UTCTime	Unsigned 8-bit integer	Unsigned 16-bit integer	Unsigned 16-bit integer
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	Signed 8-bit integer	Unsigned 8-bit integer	8-bit BitMap	Unsigned 8-bit integer	Octets (non-ZCL Data Type)
Field Name	Average Load Adjustment Percentage Applied (O)	Duty Cycle Applied (O)	Event Control (M)	Signature Type (M)[a]	Signature (O)[b]

a. CCB 1324
b. CCB 1324

Figure D.5 Format of the *Report Event Status* Command Payload

D.2.3.3.1.1.1 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 (UTCTime data type) identifying when the Load Control Event was issued.

Event Status (mandatory): Table D.9 lists the valid values returned in the Event Status field.

Table D.9 Event Status Field Values

Value	Description
0x00	Reserved for future use.
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

Table D.9 Event Status Field Values (Continued)

Value	Description
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”).
0x0B to 0xF7	Reserved for future use.
0xF8	Rejected - Invalid Cancel Command (Default)
0xF9	Rejected - Invalid Cancel Command (Invalid Effective Time)
0xFA	Reserved
0xFB	Rejected - Event was received after it had expired (Current Time > Start Time + Duration)
0xFC	Reserved for future use.
0xFD	Rejected - Invalid Cancel Command (Undefined Event)
0xFE	*Load Control Event* command Rejected
0xFF	Reserved for future use.

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[179] 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.

179.CCB 1072

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 D.2.2.3.1.1.1.

Event Control (mandatory): Identifies additional control options for the event. Refer to sub-clause D.2.2.3.1.1.1.

Signature Type (mandatory[180]): An 8-bit Unsigned integer enumerating the type of algorithm use to create the Signature. The enumerated values are:

Enumerated Value	Signature Type
0x00	No Signature[a]
0x01	ECDSA
0x02 to 0xFF	Reserved

a. CCB 1398

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.[181]

Signature (optional[182]): A non-repudiation signature created by using the Matyas-Meyer-Oseas hash function (specified in Annex B.6 in [B3]) used in conjunction with ECDSA. The signature creation process will occur in two steps:

180.CCB 1398
181.CCB 1398

1 Pass the first ten fields, which includes all fields up to the Signature field, of the *Report Event Status* command (listed in Figure D.5) 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.*

D.2.3.3.1.2 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.)

D.2.3.3.2 *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 D.2.2.3.1) will be sent covering both active and scheduled Load Control Events.

D.2.3.3.2.1 Payload Format

The *Get Scheduled Events* command payload shall be formatted as illustrated in Figure D.6

Octets	**4**	**1**
Data Type	UTCTime	Unsigned 8-bit integer
Field Name	Start Time (M)	Number of Events (M)

Figure D.6 Format of the *Get Scheduled Events* Command Payload

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.[183]

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*[184] command (EndTime would be StartTime plus Duration of the event listed in the device's event table).[185]

D.2.3.3.2.2 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.[186]

D.2.3.4 Commands Received

The client receives the cluster-specific commands detailed in sub-clause D.2.2.

D.2.3.5 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.

D.2.4 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

183.CCB 1244
184.CCB 1325
185.CCB 1244
186.CCB 1119

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.*

D.2.4.1 Load Control Rules, Server

D.2.4.1.1 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.

D.2.4.1.2 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.

D.2.4.2 Load Control Rules, Client

D.2.4.2.1 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.[187]

When ending a load control event, the load control device will support the same randomization features as provided in the start load control event.

D.2.4.2.2 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.

D.2.4.2.3 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.

D.2.4.2.4 Opt-Out Messages[188]

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

187.CCB1293
188.CCB 1339

where the event has been acknowledged and started, the override message will be sent when the override occurs.

D.2.4.2.5 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.

D.2.4.2.6 Demand Response and Load Control Transaction Examples

The following example in Figure D.7 depicts the transactions that would take place for two events, one that is successful and another that is overridden by the user.

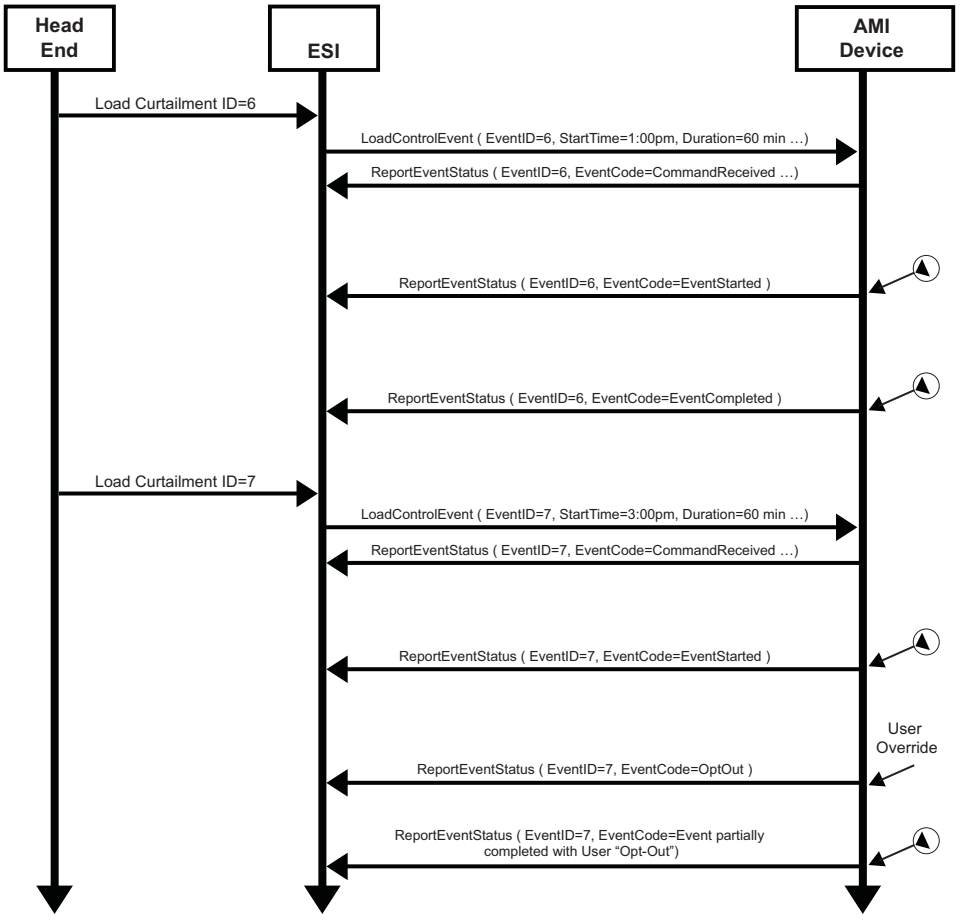

Figure D.7 Example of Both a Successful and an Overridden Load Curtailment Event

The example in Figure D.8 depicts the transactions that would take place when an event is superseded by an event that is eventually cancelled.

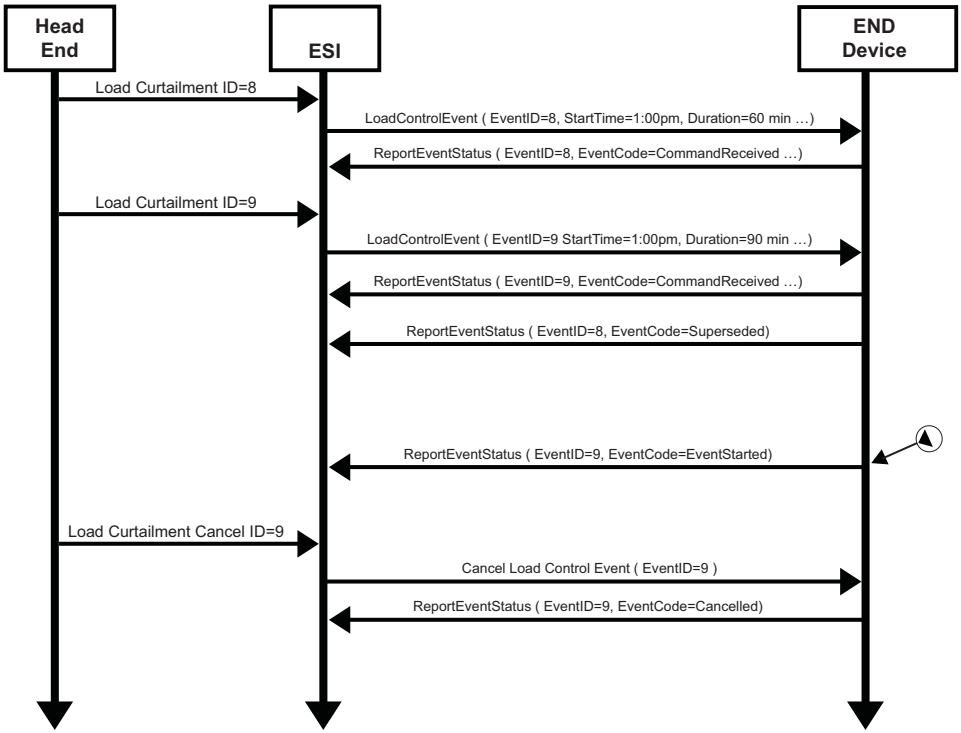

Figure D.8 Example of a Load Curtailment Superseded and Another Cancelled

Please refer to Annex E for more information regarding the management and behavior of overlapping events.

D.3 Metering[189] Cluster

D.3.1 Overview

The Metering[190] 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

189.CCB 940

190.CCB940

of sophistication. The Metering[191] 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 D.6.

The following figures identify three configurations as examples utilizing the Metering[192] Cluster.

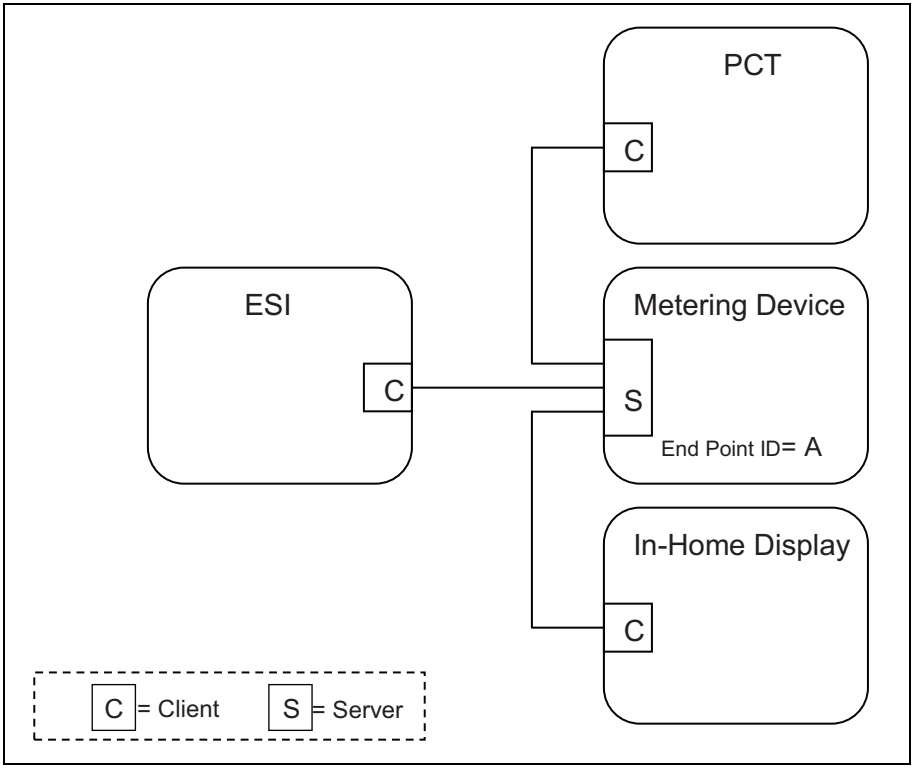

Figure D.9 Standalone ESI[193] Model with Mains Powered Metering Device

In the example shown in Figure D.9, the metering device is the source of information provided via the Metering[194] Cluster Server.

191. CCB 940
192. CCB 940
193. CCB 1072
194. CCB 940

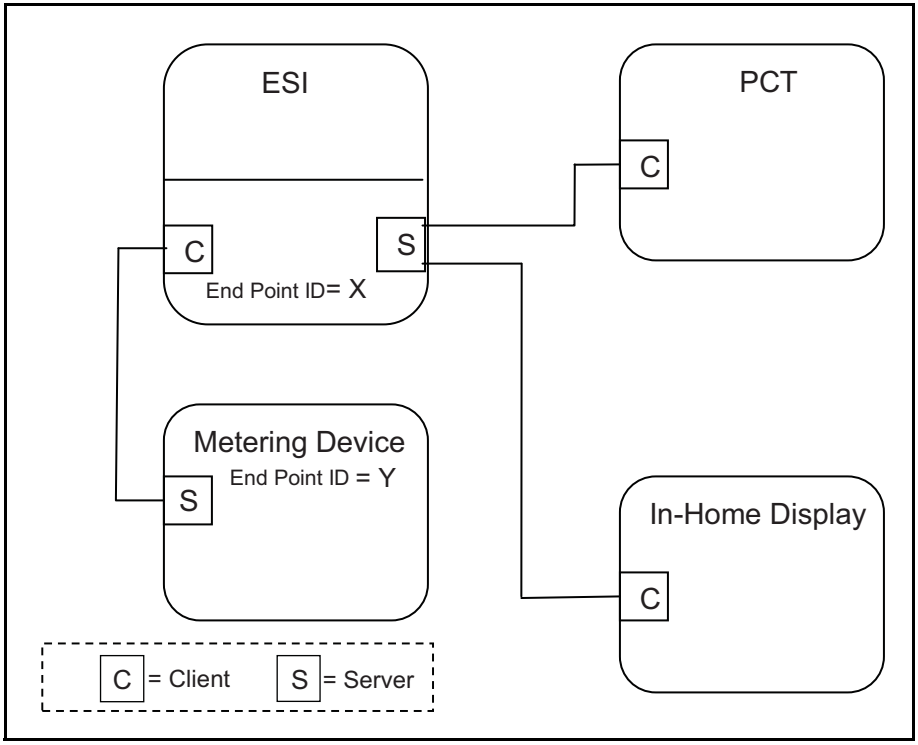

Figure D.10 Standalone ESI[195] Model with Battery Powered Metering Device

In the example shown in Figure D.10, the metering device is running on battery power and its duty cycle for providing information is unknown. It's expected the ESI[196] 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[197] Cluster).

195.CCB 1072
196.CCB 1072
197.CCB 940

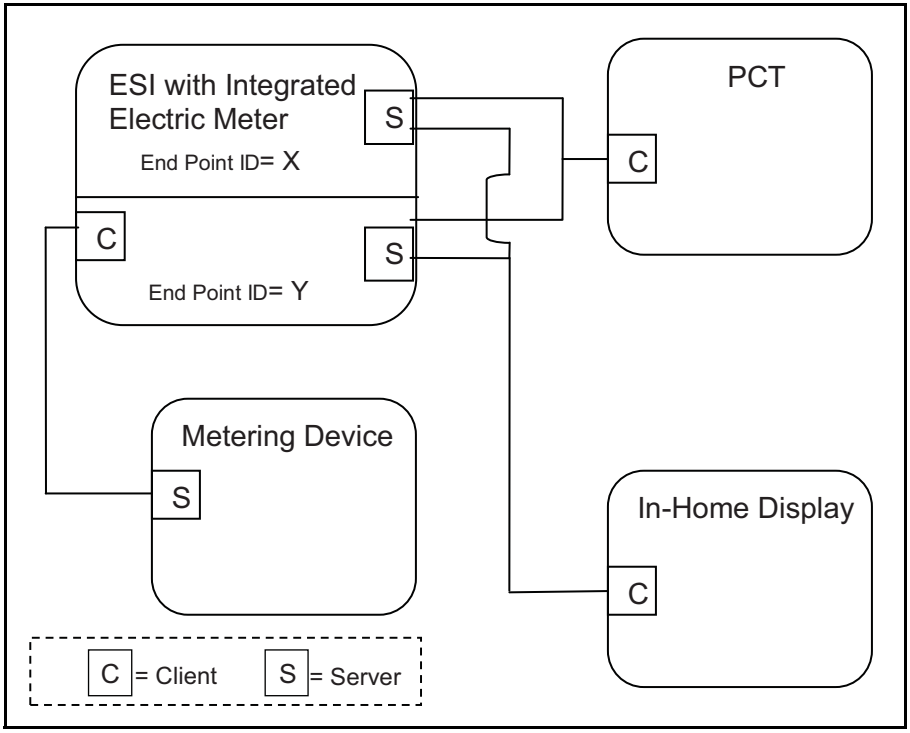

Figure D.11 ESI[198] Model with Integrated Metering Device

In the example shown in Figure D.11, much like the previous example in Figure D.10, the external metering device is running on battery power and its duty cycle for providing information is unknown. It's expected the ESI[199] 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[200] Cluster). Since the ESI[201] can also contain an integrated metering device where its information is also conveyed through the Metering[202] 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[203] cluster support by performing service discoveries. It can also identify if an Endpoint ID is a mailbox/

198.CCB 1072
199.CCB 1072
200.CCB 940
201.CCB 1072
202.CCB 940
203.CCB 1072

mirror of a metering device by reading the *MeteringDeviceType* attribute (refer to sub-clause D.3.2.2.4.7).

In the above examples (Figure D.10 and Figure D.11), it's expected the ESI[204] 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[205] will update its mailbox (EndPoint ID Y in Figure D.10 and Figure D.11) 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 D.3.3.4.[206]

Other Smart Energy devices can access EndPoint Y in the ESI[207] to receive the latest information just as they would to access information in the ESI[208]'s integrated Electric meter (as in Figure D.11, EndPoint X) and other Metering devices (as in Figure D.9, EndPoint A).

D.3.2 Server

D.3.2.1 Dependencies

Subscribed reporting of [209]Metering attributes.

D.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 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 D.10.

Note: *Certain attributes within this cluster are provisionary and not certifiable. Refer to the individual attribute sets for details of the relevant attributes.*

204.CCB 1072
205.CCB 1072
206.CCB 1219
207.CCB 1072
208.CCB 1072
209.CCB 940

Table D.10 Metering[a] Cluster Attribute Sets

Attribute Set Identifier	Description
0x00	Reading Information Set
0x01	TOU Information Set
0x02	Meter Status
0x03	Formatting
0x04	Historical Consumption[b]
0x05	Load Profile Configuration
0x06	Supply Limit[c]
0x07	Block Information[d]
0x08	Alarms[e]
0x09[f] to 0xFF	Reserved[g]

a. CCB 940
b. CCB 1015
c. CCB 974
d. Incremental Release 1
e. Incremental Release 1
f. Incremental Release 1
g. CCB 974

D.3.2.2.1 Reading Information Set

The following set of attributes 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.*

Table D.11 Reading Information Attribute Set

Identifier	Name	Type[a]	Range[b]	Access	Default	Man. /Opt.
0x00	*CurrentSummationDelivered*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	M
0x01	*CurrentSummationReceived*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x02	*CurrentMaxDemandDelivered*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x03	*CurrentMaxDemandReceived*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x04	*DFTSummation*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x05	*Daily Freeze Time*	Unsigned 16-bit Integer	0x0000 to 0x183C	Read Only	0x0000	O
0x06	*PowerFactor*	Signed 8-bit Integer	-100 to +100	Read Only	0x00	O
0x07	*ReadingSnapShotTime*	UTCTime		Read Only	-	O
0x08	*CurrentMaxDemandDeliveredTime*	UTCTime		Read Only	-	O
0x09	*CurrentMaxDemandReceivedTime*	UTCTime		Read Only	-	O
0x0A[c]	*DefaultUpdatePeriod*	Unsigned 8-bit Integer	0x00 to 0xFF	Read Only	0x1E	O
0x0B[d]	*FastPollUpdatePeriod*	Unsigned 8-bit Integer	0x00 to 0xFF	Read Only	0x05	O
0x0C[e]	*CurrentBlockPeriodConsumptionDelivered*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O

Table D.11 Reading Information Attribute Set (Continued)

Identifier	Name	Type[a]	Range[b]	Access	Default	Man. /Opt.
0x0D[f]	*DailyConsumptionTarget*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O
0x0E[g]	*CurrentBlock*	8-bit Enumeration	0x00 to 0x10	Read Only	-	O
0x0F[h]	*ProfileIntervalPeriod*	8-bit Enumeration	0x00 to 0xFF	Read Only	-	O
0x10[i]	*IntervalReadReportingPeriod*	Unsigned 16-bit Integer	0x0000 to 0xFFFF	Read Only	0	O
0x11[j]	*PresetReadingTime*	Unsigned 16-bit Integer	0x0000 to 0x173B	Read Only	0x0000	O
0x12[k]	*VolumePerReport*	Unsigned 16-bit Integer	0x0000 to 0xFFFF	Read Only	-	O
0x13[l]	*FlowRestriction*	Unsigned 8-bit Integer	0x00 to 0xFF	Read Only	-	O
0x14[m]	*Supply Status*	8-bit Enumeration[n]	0x00 to 0xFF	Read Only	-	O
0x15[o]	*CurrentInletEnergyCarrierSummation*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	M:Heat M:Cooling O:others
0x16[p]	*CurrentOutletEnergyCarrierSummation*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x17[q]	*InletTemperature*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	M:Heat M:Cooling O:others

Table D.11 Reading Information Attribute Set (Continued)

Identifier	Name	Type[a]	Range[b]	Access	Default	Man. /Opt.
0x18[r]	*OutletTemperature*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	M:Heat M:Cooling O:others
0x19[s]	*ControlTemperature*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O
0x1A[t]	*CurrentInletEnergyCarrierDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O
0x1B[u]	*CurrentOutletEnergyCarrierDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O
0x1C[v]	*PreviousBlockPeriodConsumptionDelivered*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x1D to 0xFF[w]	Reserved					

a. CCB 1270
b. CCB 1270
c. Incremental Release 1
d. Incremental Release 1
e. Incremental Release 1
f. Incremental Release 1
g. Incremental Release 1
h. Incremental Release 1
i. Incremental Release 1
j. Incremental Release 1
k. Incremental Release 1
l. Incremental Release 1
m. Incremental Release 1
n. CCB 1384
o. Incremental Release 1
p. Incremental Release 1
q. Incremental Release 1
r. Incremental Release 1
s. Incremental Release 1

t. Incremental Release 1
u. Incremental Release 1
v. CCB 1500
w. CCB 1500

D.3.2.2.1.1 *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.

D.3.2.2.1.2 *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.

D.3.2.2.1.3 *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.

D.3.2.2.1.4 *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.

D.3.2.2.1.5 *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 D.3.2.2.1.6.

D.3.2.2.1.6 *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).

D.3.2.2.1.7 *PowerFactor* Attribute

PowerFactor contains the Average Power Factor ratio in 1/100ths. Valid values are 0 to 99.

D.3.2.2.1.8 *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.

D.3.2.2.1.9 *CurrentMaxDemandDeliveredTime* Attribute

The *CurrentMaxDemandDeliveredTime* attribute represents the time when *CurrentMaxDemandDelivered* reading was captured.

D.3.2.2.1.10 *CurrentMaxDemandReceivedTime* Attribute

The *CurrentMaxDemandReceivedTime* attribute represents the time when *CurrentMaxDemandReceived* reading was captured.

D.3.2.2.1.11 *DefaultUpdatePeriod* Attribute[210]

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.

D.3.2.2.1.12 *FastPollUpdatePeriod* Attribute[211]

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

210. Incremental Release 1
211. Incremental Release 1

FastPollUpdatePeriod. The *FastPollUpdatePeriod* may apply to other attributes as defined by the device manufacturer.

D.3.2.2.1.13 *CurrentBlockPeriodConsumptionDelivered* Attribute[212]

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.

D.3.2.2.1.14 *DailyConsumptionTarget* Attribute[213]

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.

D.3.2.2.1.15 *CurrentBlock* Attribute[214]

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.

212. Incremental Release 1
213. Incremental Release 1
214. Incremental Release 1

Table D.12 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
0x11 to 0xFF	Reserved

D.3.2.2.1.16 *ProfileIntervalPeriod* Attribute[215]

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 D.3.2.3.1.1.1.

D.3.2.2.1.17 *IntervalReadReportingPeriod* Attribute[216]

The *IntervalReadReportingPeriod* attribute represents how often (in minutes) the water or gas meter is to wake up and provide interval data. E.g.: If

215. Incremental Release 1

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.[217]

D.3.2.2.1.18 PresetReadingTime[218]

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.

D.3.2.2.1.19 *VolumePerReport* Attribute[219]

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.

D.3.2.2.1.20 *FlowRestriction* Attribute[220]

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.

D.3.2.2.1.21 *SupplyStatus* Attribute[221]

The *SupplyStatus* attribute represents the state of the supply at the customer's premises. The enumerated values for this field are outlined in Table D.13:

216. Incremental Release 1
217. Incremental Release 1
218. Incremental Release 1
219. Incremental Release 1
220. Incremental Release 1
221. Incremental Release 1

Table D.13 Supply Status Attribute Enumerations

Enumerated Value	Status
0x00	Supply OFF
0x01	Supply OFF/ARMED
0x02	Supply ON
0x03 to 0xFF	Reserved for future use

D.3.2.2.1.22 *CurrentInletEnergyCarrierSummation* Attribute[222]

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 D.21).

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.

D.3.2.2.1.23 *CurrentOutletEnergyCarrierSummation* Attribute[223]

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 D.21).

D.3.2.2.1.24 *InletTemperature* Attribute[224]

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 D.21).

222. Incremental Release 1
223. Incremental Release 1
224. Incremental Release 1

D.3.2.2.1.25 *OutletTemperature* Attribute[225]

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 D.21).

D.3.2.2.1.26 *ControlTemperature* Attribute[226]

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 D.21).

D.3.2.2.1.27 *CurrentInletEnergyCarrierDemand* Attribute[227]

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 D.21).

For a heat or cooling meter this will be the current absolute flow rate measured on the inlet.

D.3.2.2.1.28 *CurrentOutletEnergyCarrierDemand* Attribute[228]

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 D.21).

For a heat or cooling meter this will be the current absolute flow rate measured on the outlet.

225. Incremental Release 1
226. Incremental Release 1
227. Incremental Release 1
228. Incremental Release 1

D.3.2.2.1.29 *PreviousBlockPeriodConsumptionDelivered* Attribute[229]

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.

D.3.2.2.2 Summation TOU Information Set

The following set of attributes 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 D.14 TOU Information Attribute Set

0x00	*CurrentTier1SummationDelivered*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x01	*CurrentTier1SummationReceived*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x02	*CurrentTier2SummationDelivered*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x03	*CurrentTier2SummationReceived*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x04	*CurrentTier3SummationDelivered*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x05	*CurrentTier3SummationReceived*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x06	*CurrentTier4SummationDelivered*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x07	*CurrentTier4SummationReceived*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O

229.CCB 1500

Table D.14 TOU Information Attribute Set (Continued)

0x08	*CurrentTier5SummationDelivered*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x09	*CurrentTier5SummationReceived*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x0A	*CurrentTier6SummationDelivered*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x0B	*CurrentTier6SummationReceived*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x0C[a]	*CurrentTier7SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x0D[b]	*CurrentTier7SummationReceived*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x0E[c]	*CurrentTier8SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x0F[d]	*CurrentTier8SummationReceived*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x10[e]	*CurrentTier9SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x11[f]	*CurrentTier9SummationReceived*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x12[g]	*CurrentTier10SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x13[h]	*CurrentTier10SummationReceived*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x14[i]	*CurrentTier11SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x15[j]	*CurrentTier11SummationReceived*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O

Table D.14 TOU Information Attribute Set (Continued)

0x16[k]	*CurrentTier12SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x17[l]	*CurrentTier12SummationReceived*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x18[m]	*CurrentTier13SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x19[n]	*CurrentTier13SummationReceived*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x1A[o]	*CurrentTier14SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x1B[p]	*CurrentTier14SummationReceived*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x1C[q]	*CurrentTier15SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x1D[r]	*CurrentTier15SummationReceived*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x1E to 0xFF[s]	Reserved					

a. Incremental Release 1
b. Incremental Release 1
c. Incremental Release 1
d. Incremental Release 1
e. Incremental Release 1
f. Incremental Release 1
g. Incremental Release 1
h. Incremental Release 1
i. Incremental Release 1
j. Incremental Release 1
k. Incremental Release 1
l. Incremental Release 1
m. Incremental Release 1
n. Incremental Release 1
o. Incremental Release 1

p. Incremental Release 1
q. Incremental Release 1
r. Incremental Release 1
s. Incremental Release 1

D.3.2.2.2.1 *CurrentTierNSummationDelivered* Attributes

Attributes *CurrentTier1SummationDelivered* through *CurrentTierNSummationDelivered*[230] 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.

D.3.2.2.2.2 *CurrentTierNSummationReceived* Attributes

Attributes *CurrentTier1SummationReceived* through *CurrentTierNSummationReceived*[231] 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.

D.3.2.2.3 Meter Status Attribute Set

The Meter Status Attribute Set is defined in Table D.15.

Table D.15 Meter Status Attribute Set

Identifier	Name	Type	Range	Access	Default	Man. / Opt.
0x00	*Status*	8-bit BitMap	0x00 to 0xFF	Read Only	0x00	M
0x01[a]	*Remaining BatteryLife*	Unsigned 8-bit Integer	0x00 to 0xFF	Read Only	-	O
0x02[b]	*HoursInOperation*	Unsigned 24bit Integer	0x000000 to 0xFFFFFF	Read Only	-	M:Heat M:Cooling O:others
0x03[c]	*HoursInFault*	Unsigned 24bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O
0x04-0xFF[d]	Reserved					

a. Incremental Release 1

b. Incremental Release 1

230. Incremental Release 1

231. Incremental Release 1

c. Incremental Release 1

d. Incremental Release 1

D.3.2.2.3.1 *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. Tables D.16, D.17, D.18, and D.19 below 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 D.16 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, self check error.

Table D.17 Meter *Status* Attribute (Gas)[a]

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

a. Incremental Release 1

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 D.18 Meter *Status* Attribute (Water)[a]

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

a. Incremental Release 1

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 D.19 Meter *Status* Attribute (Heat and Cooling)[a]

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[b]

a. CCB 1181
b. Incremental Release 1

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.*

D.3.2.2.3.2 *RemainingBatteryLife* Attribute[232]

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.

D.3.2.2.3.3 *HoursInOperation* Attribute[233]

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.*

D.3.2.2.3.4 *HoursInFault* Attribute[234]

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.*

D.3.2.2.4 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 = Summation 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

232. Incremental Release 1
233. Incremental Release 1
234. Incremental Release 1

listed in sub-clauses D.3.2.2.4.4, D.3.2.2.4.5, D.3.2.2.4.6, D.3.2.2.4.11, D.3.2.2.4.12, and D.3.2.2.4.14.

The following table shows examples that demonstrate the relation between these attributes.[235]

Table D.20 Formatting Examples

Attribute	Example 1	Example 2	Example 3
Value as transmitted and received	52003	617	23629
UnitofMeasure	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 D.21.

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 D.21 Formatting Attribute Set

Identifier	Name	Type	Range[a]	Access	Default	Man./ Opt.
0x00	*UnitofMeasure*	8-bit Enumeration	0x00 to 0xFF	Read Only	0x00	M
0x01	*Multiplier*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O
0x02	*Divisor*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O
0x03	*SummationFormatting*	8-bit BitMap	0x00 to 0xFF	Read Only	-	M

235.CCB 1269

Table D.21 Formatting Attribute Set (Continued)

Identifier	Name	Type	Range[a]	Access	Default	Man./ Opt.
0x04	*DemandFormatting*	8-bit BitMap	0x00 to 0xFF	Read Only	-	O
0x05	*HistoricalConsumptionFormatting*	8-bit BitMap	0x00 to 0xFF	Read Only	-	O
0x06	*MeteringDeviceType*	8-bit BitMap	0x00 to 0xFF	Read Only	-	M
0x07[b]	*SiteID*	Octet String	1 to 33 Octets	Read only	-	O
0x08[c]	*MeterSerialNumber*	Octet String	1 to 25 Octets	Read only	-	O
0x09[d]	*EnergyCarrierUnitOfMeasure*	8-bit Enumeration	0x00 to 0xFF	Read Only	-	M:Heat M:Cooling O:others
0x0A[e]	*EnergyCarrierSummationFormatting*	8-bit BitMap	0x00 to 0xFF	Read Only	-	M:Heat M:Cooling O:others
0x0B[f]	*EnergyCarrierDemandFormatting*	8-bit BitMap	0x00 to 0xFF	Read Only	-	O
0x0C[g]	*TemperatureUnitOfMeasure*	8-bit Enumeration	0x00 to 0xFF	Read Only	-	M:Heat M:Cooling O:others
0x0D[h]	*TemperatureFormatting*	8-bit BitMap	0x00 to 0xFF	Read Only	-	M:Heat M:Cooling O:others
0x0E to 0xFF[i]	Reserved					

a. CCB 1292

b. Incremental Release 1

c. Incremental Release 1

d. Incremental Release 1

e. Incremental Release 1

f. Incremental Release 1

g. Incremental Release 1

h. Incremental Release 1

i. Incremental Release 1

D.3.2.2.4.1 *UnitofMeasure* Attribute

UnitofMeasure provides a label for the Energy, Gas, or Water being measured by the metering device. The unit 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 D.22.

Table D.22 *UnitofMeasure* Attribute Enumerations

Values	Description
0x00	kWh (Kilowatt Hours) & kW (Kilowatts) in pure binary format[a]
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[b]
0x0B	Unitless in pure binary format[c]
0x0C	MJ (Mega Joule) and MJ/s (Mega Joule per second (MW)) in pure binary format[d]
0x0D to 0x7F	Reserved for future use.
0x80	kWh (Kilowatt Hours) & kW (Kilowatts) in BCD format[e]
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
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

Table D.22 *UnitofMeasure* Attribute Enumerations (Continued)

Values	Description
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[f]
0x8B	unitless in BCD format[g]
0x8C	MJ (Mega Joule) and MJ/s (Mega Joule per second (MW)) in BCD format[h]
0x8D to 0xFF	Reserved for future use.

a. CCB 1341
b. CCB 1124
c. CCB 1170
d. CCB 1180
e. CCB 1341
f. CCB 1124
g. CCB 1170
h. CCB 1180

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.*

D.3.2.2.4.2 *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.

D.3.2.2.4.3 *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.

D.3.2.2.4.4 *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

D.3.2.2.4.5 *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*

D.3.2.2.4.6 *HistoricalConsumptionFormatting* Attribute

HistoricalConsumptionFormatting provides a method to properly decipher the number of digits and the decimal location of the values found in the Historical[236] 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*

D.3.2.2.4.7 *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 D.23. (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.)[237]

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).[238] 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 D.23 *MeteringDeviceType* Attribute

Values	Description
0	Electric Metering
1	Gas Metering
2	Water Metering
3	Thermal Metering (deprecated)[a]

236.CCB 1015
237.CCB1198
238.CCB 1376

Table D.23 *MeteringDeviceType* Attribute (Continued)

Values	Description
4	Pressure Metering
5	Heat Metering[b]
6	Cooling Metering[c]
7[d] to 127	Reserved for future growth
128	Mirrored Gas Metering
129	Mirrored Water Metering
130	Mirrored Thermal Metering (deprecated)[e]
131	Mirrored Pressure Metering
132	Mirrored Heat Metering[f]
133	Mirrored Cooling Metering[g]
134 to 255	Reserved for future growth

a. CCB 986
b. CCB 986
c. CCB 986
d. CCB 986
e. CCB 986
f. CCB 986
g. CCB 986

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.*[239]

D.3.2.2.4.8 *SiteID* Attribute[240]

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

239. CCB 986
240. Incremental Release 1

numeric but is defined for the possibility of an alpha-numeric format by specifying an octet string.

D.3.2.2.4.9 *MeterSerialNumber* Attribute[241]

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.

D.3.2.2.4.10 *EnergyCarrierUnitOfMeasure* Attribute[242]

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 D.22).

D.3.2.2.4.11 *EnergyCarrierSummationFormatting* Attribute[243]

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*

D.3.2.2.4.12 *EnergyCarrierDemandFormatting* Attribute[244]

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.

241. Incremental Release 1
242. CCB 1181
243. CCB 1181
244. CCB 1181

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*

D.3.2.2.4.13 *TemperatureUnitOfMeasure* Attribute[245]

The *TemperatureUnitOfMeasure* specifies the unit of measure that temperatures are measured in. The enumeration of this attribute is as follows.

Table D.24 *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.
0x03-0x7F	Reserved for future use
0x80	K (Degrees Kelvin) in BCD format.
0x81	°C (Degrees Celsius) in BCD format.
0x82	°F (Degrees Fahrenheit) in BCD format.
0x83-0xFF	Reserved for future use

D.3.2.2.4.14 *TemperatureFormatting* Attribute[246]

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:

245.CCB 1181

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*

D.3.2.2.5 *Historical Consumption* Attribute[247]

The Historical Attribute Set is defined in Table D.25.[248]

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 D.25 Historical Attribute Set [a]

Identifier	Name	Type	Range	Access	Default	Man./ Opt.
0x00	*InstantaneousDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607[b]	Read Only	0x00	O
0x01	*CurrentDayConsumptionDelivered*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O
0x02	*CurrentDayConsumptionReceived*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O
0x03	*PreviousDayConsumptionDelivered*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O
0x04	*PreviousDayConsumptionReceived*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O

246.CCB 1181
247.CCB 1015
248.CCB 1015

Table D.25 Historical Attribute Set (Continued)[a]

Identifier	Name	Type	Range	Access	Default	Man./ Opt.
0x05	*CurrentPartialProfileIntervalStartTimeDelivered*	UTCTime		Read Only	-	O
0x06	*CurrentPartialProfileIntervalStartTimeReceived*	UTCTime	0x000000 to 0xFFFFFF	Read Only	-	O
0x07	*CurrentPartialProfileIntervalValueDelivered*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read Only	-[c]	O[d]
0x08	*CurrentPartialProfileIntervalValueReceived*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read Only	-[e]	O[f]
0x09[g]	*CurrentDayMaxPressure*	Unsigned 48-bit Integer	0x000000 000000 to 0xFFFFFF FFFFFF	Read Only	-	O
0x0A[h]	*CurrentDayMinPressure*	Unsigned 48-bit Integer	0x000000 000000 to 0xFFFFFF FFFFFF	Read Only	-	O
0x0B[i]	*PreviousDayMaxPressure*	Unsigned 48-bit Integer	0x000000 000000 to 0xFFFFFF FFFFFF	Read Only	-	O
0x0C[j]	*PreviousDayMinPressure*	Unsigned 48-bit Integer	0x000000 000000 to 0xFFFFFF FFFFFF	Read Only	-	O
0x0D[k]	*CurrentDayMaxDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O
0x0E[l]	*PreviousDayMaxDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O
0x0F[m]	*CurrentMonthMaxDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O
0x10[n]	*CurrentYearMaxDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O

Table D.25 Historical Attribute Set (Continued)[a]

Identifier	Name	Type	Range	Access	Default	Man./ Opt.
0x11[o]	*CurrentDayMaxEnergyCarrierDemand*	Signed 24-bit integer	-8,388,607 to 8,388,607	Read Only	-	O
0x12[p]	*PreviousDayMaxEnergyCarrierDemand*	Signed 24-bit integer	-8,388,607 to 8,388,607	Read Only	-	O
0x13[q]	*CurrentMonthMaxEnergyCarrierDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O
0x14[r]	*CurrentMonthMinEnergyCarrierDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O
0x15[s]	*CurrentYearMaxEnergyCarrierDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O
0x16[t]	*CurrentYearMinEnergyCarrierDemand*	Signed 24-bit Integer	-8,388,607 to 8,388,607	Read Only	-	O
0x17 to 0xFF[u]	Reserved					

a. CCB 1015
b. CCB 1082
c. CCB 982
d. CCB 982
e. CCB 982
f. CCB 982
g. Incremental Release 1
h. Incremental Release 1
i. Incremental Release 1
j. Incremental Release 1
k. Incremental Release 1
l. Incremental Release 1
m. Incremental Release 1
n. Incremental Release 1
o. Incremental Release 1
p. Incremental Release 1
q. Incremental Release 1
r. Incremental Release 1
s. Incremental Release 1

t. Incremental Release 1

u. Incremental Release 1

D.3.2.2.5.1 *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.

D.3.2.2.5.2 *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.

D.3.2.2.5.3 *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.

D.3.2.2.5.4 *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.

D.3.2.2.5.5 *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.

D.3.2.2.5.6 *CurrentPartialProfileIntervalStartTimeDelivered* Attribute

CurrentPartialProfileIntervalStartTimeDelivered represents the start time of the current Load Profile interval being accumulated for commodity delivered.

D.3.2.2.5.7 *CurrentPartialProfileIntervalStartTimeReceived* Attribute

CurrentPartialProfileIntervalStartTimeReceived represents the start time of the current Load Profile interval being accumulated for commodity received.

D.3.2.2.5.8 *CurrentPartialProfileIntervalValueDelivered* Attribute

CurrentPartialProfileIntervalValueDelivered represents the value of the current Load Profile interval being accumulated for commodity delivered.

D.3.2.2.5.9 *CurrentPartialProfileIntervalValueReceived* Attribute

CurrentPartialProfileIntervalValueReceived represents the value of the current Load Profile interval being accumulated for commodity received.

D.3.2.2.5.10 *CurrentDayMaxPressure* Attribute

CurrentDayMaxPressure is the maximum pressure reported during a day from the water or gas meter.

D.3.2.2.5.11 *PreviousDayMaxPressure* Attribute

PreviousDayMaxPressure represents the maximum pressure reported during previous day from the water or gas meter.

D.3.2.2.5.12 *CurrentDayMinPressure* Attribute

CurrentDayMinPressure is the minimum pressure reported during a day from the water or gas meter.

D.3.2.2.5.13 *PreviousDayMinPressure* Attribute

PreviousDayMinPressure represents the minimum pressure reported during previous day from the water or gas meter.

D.3.2.2.5.14 *CurrentDayMaxDemand* Attribute

CurrentDayMaxDemand represents the maximum demand or rate of delivered value of Energy, Gas, or Water being utilized at the premises.

D.3.2.2.5.15 *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.

D.3.2.2.5.16 *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.

D.3.2.2.5.17 *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.

D.3.2.2.5.18 *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.

D.3.2.2.5.19 *PreviousDayMaxEnergyCarrierDemand* Attribute

PreviousDayMaxEnergyCarrierDemand is the maximum energy carrier demand reported during the previous day from the meter.

D.3.2.2.5.20 *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.

D.3.2.2.5.21 *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.*

D.3.2.2.5.22 *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.

D.3.2.2.5.23 *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*

D.3.2.2.6 Load Profile Configuration

The Load Profile Configuration Attribute Set is defined in Table D.26.

Table D.26 Load Profile Configuration Attribute Set

Identifier	Name	Type	Range	Access	Default	Man./ Opt.
0x00	*MaxNumberOfPeriodsDelivered*	Unsigned 8 bit Integer	0x00 to 0xFF	Read Only	0x18	O
0x01 to 0xFF	Reserved					

D.3.2.2.6.1 *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 sub-clause 5.3.8 for further details on Fragmentation settings.[249]

D.3.2.2.7 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[250] 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 D.27.

Table D.27 Supply Limit Attribute Set

Identifier	Name	Type	Range	Access	Default	Man / Opt
0x00	*CurrentDemandDelivered*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read only		O
0x01	*DemandLimit*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read only		O
0x02	*DemandIntegrationPeriod*	Unsigned 8-bit Integer	0x01 to 0xFF	Read only	-	O
0x03	*NumberOfDemandSubintervals*	Unsigned 8-bit Integer	0x01 to 0xFF	Read only	-	O
0x04 - 0xFF	Reserved					

D.3.2.2.7.1 *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.

249. CCB 983
250. Incremental Release 1/CCB 1570

D.3.2.2.7.2 *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.[251]

D.3.2.2.7.3 *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.

D.3.2.2.7.4 *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 the each of 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[252].

D.3.2.2.8 Block Information Set[253]

The following set of attributes 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.

251. CCB 1103
252. CCB 974
253. Incremental Release 1

Table D.28 Block Information Attribute Set[a]

Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0x00	*CurrentNoTierBlock1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x01	*CurrentNoTierBlock2SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x02	*CurrentNoTierBlock3SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x0N	… *CurrentNoTierBlockN+1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x0F	*CurrentNoTierBlock16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x10	*CurrentTier1Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x11	*CurrentTier1Block2SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x12	*CurrentTier1Block3SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x1N	*CurrentTier1BlockN+1SummationDelivered* …	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x1F	*CurrentTier1Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O

Table D.28 Block Information Attribute Set[a] (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0x20	*CurrentTier2 Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x2N	*CurrentTier2 BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x2F	*CurrentTier2 Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x30	*CurrentTier3 Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x3N	*CurrentTier3 BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x3F	*CurrentTier3 Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x40	*CurrentTier4 Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x4N	*CurrentTier4 BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x4F	*CurrentTier4 Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x50	*CurrentTier5 Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O

Table D.28 Block Information Attribute Set[a] (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0x5N	*CurrentTier5 BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x5F	*CurrentTier5 Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x60	*CurrentTier6 Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x6N	*CurrentTier6 BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x6F	*CurrentTier6 Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x70	*CurrentTier7 Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x7N	*CurrentTier7 BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x7F	*CurrentTier7 Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x80	*CurrentTier8 Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x8N	*CurrentTier8 BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O

Table D.28 Block Information Attribute Set[a] (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0x8F	*CurrentTier8Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x90	*CurrentTier9Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x9N	*CurrentTier9BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0x9F	*CurrentTier9Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xA0	*CurrentTier10Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xAN	*CurrentTier10BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xAF	*CurrentTier10Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xB0	*CurrentTier11Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xBN	*CurrentTier11BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xBF	*CurrentTier11Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O

Table D.28 Block Information Attribute Set[a] (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0xC0	*CurrentTier12Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xCN	*CurrentTier12BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xCF	*CurrentTier12Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xD0	*CurrentTier13Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xDN	*CurrentTier13BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xDF	*CurrentTier13Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xE0	*CurrentTier14Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xEN	*CurrentTier14BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xEF	*CurrentTier14Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O

Table D.28 Block Information Attribute Set[a] (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0xF0	*CurrentTier15Block1SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O
0xFN	*CurrentTier15BlockN+1SummationDelivered …*	Unsigned 48-bit integer	0x000000000000 to 0Xffffffffffff	Read only	-	O
0xFF	*CurrentTier15Block16SummationDelivered*	Unsigned 48-bit integer	0x000000000000 to 0xFFFFFFFFFFFF	Read only	-	O

a. Incremental Release 1

D.3.2.2.8.1 *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.*

D.3.2.2.9 Alarms Set[254]

The following set of attributes provides a means to control which alarms may be generated from the meter.

Note: *Alarms Attribute Set 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.*

254.CCB 1179

Table D.29 Alarm Attribute Set

Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0x00	Generic AlarmMask	16-bit BitMap	0x0000 - 0xffff	Read/ Write	0xffff	O
0x01	Electricity AlarmMask	32-bit BitMap	0x00000000 - 0xffffffff	Read/ Write	0xffffffff	O
0x02	Generic Flow/ Pressure AlarmMask	16-bit BitMap	0x0000 - 0xffff	Read/ Write	0xffff	O
0x03	Water Specific AlarmMask	16-bit BitMap	0x0000 - 0xffff	Read/ Write	0xffff	O
0x04	Heat and Cooling Specific AlarmMask	16-bit BitMap	0x0000 - 0xffff	Read/ Write	0xffff	O
0x05	Gas Specific AlarmMask	16-bit BitMap	0x0000 - 0xffff	Read/ Write	0xffff	O

D.3.2.2.9.1 *AlarmMask* Attributes

The *AlarmMask* attributes of the Alarm Attribute Set specify whether each of the alarms listed in the corresponding alarm group in Table D.30 through Table D.36 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.

D.3.2.2.9.2 Alarm Codes

The alarm codes are organized in logical groups corresponding to the meter type as listed below. 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 D.30 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
70-FF	**Reserved**

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 D.3.2.2.3. 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.

Table D.31 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

Table D.31 Generic Alarm Group (Continued)

Alarm Code	Alarm Condition
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 below.

Table D.32 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
1A	Frequency too High L1
1B	Frequency too Low L2
1C	Frequency too High L2
1D	Frequency too Low L3
1E	Frequency too High L3
1F	Ground Fault
20	Electric Tamper Detect
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 below.

Table D.33 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 below.

Table D.34 Water Specific Alarm Group

Alarm Code	Alarm Condition
40-4F	Reserved

The Heat and Cooling Specific Alarm Group defines alarms specific for Heat or Cooling meters as defined below.

Table D.35 Heat and Cooling Specific Alarm Group

Alarm Code	Alarm Condition
50	Inlet Temperature Sensor Fault
51	Outlet Temperature Sensor Fault
52-5F	Reserved

The Gas Specific Alarm Group defines alarms specific for Gas meters as defined below.

Table D.36 Gas Specific Alarm Group

Alarm Code	Alarm Condition
60-6F	Reserved

D.3.2.3 Server Commands

D.3.2.3.1 Commands Generated

The command IDs generated by the Metering[255] server cluster are listed in Table D.37.

Table D.37 Generated Command IDs for the Metering[a] Server

Command Identifier Field Value	Description	Mandatory / Optional
0x00	*Get Profile Response*	O
0x01	*Request Mirror*	O
0x02	*Remove Mirror*	O
0x03[b]	*Request Fast Poll Mode Response*	O
0x04 – 0xff	Reserved	

a. CCB 940

b. Incremental Release 1

D.3.2.3.1.1 *Get Profile Response* Command

D.3.2.3.1.1.1 Payload Format

The *Get Profile Response* command payload shall be formatted as illustrated in Figure D.12.

255.CCB 940

Octets	4	1	1	1	Variable
Data Type	UTC Time	8-bit Enumeration	8-bit Enumeration	Unsigned 8-bit Integer	Series of Unsigned 24-bit Integers
Field Name	EndTime	Status	ProfileIntervalPeriod	NumberOfPeriodsDelivered	Intervals

Figure D.12 Format of the *Get Profile Response* Command Payload

D.3.2.3.1.1.2 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 D.38 lists the valid values returned in the Status field.

Table D.38 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
0x06 to 0xFF	Reserved for future use

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 following timeframes listed in Table D.39:

Table D.39 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
8 to 255	Reserved

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.

D.3.2.3.1.1.3 When Generated

This command is generated when the Client command *GetProfile* is received. Please refer to sub-clause D.3.2.4.1.1.

D.3.2.3.1.2 *Request Mirror* Command

This command is used to request the ESI[256] to mirror Metering Device data.

D.3.2.3.1.2.1 Payload Details

There are no fields for this command.

256.CCB 1072

D.3.2.3.1.2.2 Effect on Receipt

On receipt of this command, the Server shall send a *RequestMirrorReponse* command (see sub-clause D.3.2.4.1.2).

D.3.2.3.1.3 *Remove Mirror* Command

This command is used to request the ESI[257] 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.[258]

D.3.2.3.1.3.1 Payload Details

There are no fields for this command.

D.3.2.3.1.3.2 Effect on Receipt

On receipt of this command, the Server shall send a *MirrorRemoved* command (see sub-clause D.3.2.4.1.3).

D.3.2.3.1.4 *Request Fast Poll Mode Response* Command[259]

D.3.2.3.1.4.1 Payload Format

The *Request Fast Poll Mode Response* command payload shall be formatted as illustrated in Figure D.13:

Octets	1	4
Data Type	Unsigned 8-bit Integer	UTCTime
Field Name	Applied Update Period (seconds) (M)	Fast Poll Mode End Time (M)

Figure D.13 Format of the *Request Fast Poll Mode Response* Command Payload

257. CCB 1072
258. CCB 1440
259. Incremental Release 1

D.3.2.3.1.4.2 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[260].

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.

D.3.2.3.1.4.3 When Generated

This command is generated when the client command *Request Fast Poll Mode* is received.

D.3.2.3.1.4.4 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.[261]

D.3.2.4 Client Commands

D.3.2.4.1 Commands Generated

The command IDs generated by the Metering[262] client cluster are listed in Table D.40.

260.CCB1320
261.Incremental Release 1
262.CCB 940

Table D.40 Generated Command IDs for the Metering[a] Client

Command Identifier Field Value	Description	Mandatory / Optional
0x00	*Get Profile*	O
0x01	*Request Mirror Response*	O
0x02	*Mirror Removed*	O
0x03[b]	*Request Fast Poll Mode*	O
0x04-0xFF	Reserved	

a. CCB 940

b. Incremental Release 1

D.3.2.4.1.1 *Get Profile* Command

The *Get Profile* command payload shall be formatted as illustrated in Figure D.14.

Octets	1	4	1
Data Type	8-bit Enumeration[a]	UTCTime	Unsigned 8-bit Integer
Field Name	Interval Channel	End Time	NumberOfPeriods

a. CCB 1077

Figure D.14 Format of the *Get Profile* Command Payload

D.3.2.4.1.1.1 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 D.41.

Table D.41 Interval Channel Values

Enumerated Value[a]	Description
0	Consumption Delivered
1	Consumption Received
2 to 255	Not used

a. CCB 1077

EndTime: 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 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 available for the time period, only those available are returned.

D.3.2.4.1.1.2 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.

D.3.2.4.1.1.3 Command Processing Response

If failure occurs in recognizing or processing the payload of the *GetProfile* command, the appropriate enumerated ZCL status (as referenced in the ZCL Cluster Library specification) will be returned. On success, a non-Default Response is returned without a ZCL status code.[263]

D.3.2.4.1.1.4 Effect on Receipt

On receipt of this command, the device shall send a *GetProfileReponse* command (see sub-clause D.3.2.3.1.1).

D.3.2.4.1.2 *Request Mirror Response* Command

The *Request Mirror Response* Command allows the ESI[264] to inform a sleepy Metering Device it has the ability to store and mirror its data.

263.CCB 1294
264.CCB 1072

D.3.2.4.1.2.1 Payload Format

The *Request Mirror Response* command payload shall be formatted as illustrated in Figure D.15

Octets	2
Data Type	Unsigned 16-bit Integer
Field Name	EndPoint ID

Figure D.15 Format of the *Request Mirror Response* Command Payload

D.3.2.4.1.2.2 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.[265] If the ESI[266] 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.

D.3.2.4.1.3 *Mirror Removed* Command

The *Mirror Removed* Command allows the ESI[267] to inform a sleepy Metering Device mirroring support has been removed or halted.

D.3.2.4.1.3.1 Payload Format

The *Mirror Removed* command payload shall be formatted as illustrated in Figure D.16:

Octets	2
Data Type	Unsigned 16-bit Integer
Field Name	Removed EndPoint ID

Figure D.16 Format of the *Mirror Removed* Command Payload

265.CCB 1440
266.CCB 1072
267.CCB 1072

D.3.2.4.1.3.2 Payload Details

Removed EndPoint ID: 16 Bit Unsigned Integer indicating the End Point ID previously containing the Metering Devices meter data.

D.3.2.4.1.4 *Request Fast Poll Mode* Command

D.3.2.4.1.4.1 Payload Format

The *Request Fast Poll Mode* shall be formatted as illustrated in Figure D.17:

Octets	1	1
Data Type	Unsigned 8-bit Integer	Unsigned 8-bit Integer
Field Name	Fast Poll Update Period (seconds)	Duration (minutes)

Figure D.17 Format of the *Request Fast Poll Mode* Command Payload

D.3.2.4.1.4.2 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 D.3.3.2.

D.3.2.4.1.4.3 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).

D.3.2.4.1.4.4 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*.[268]

268.CCB 940

D.3.3 Metering[269] Application Guidelines

D.3.3.1 Attribute Reporting

Attribute reporting may be used for sending information in the Reading Information, TOU Information, Meter Status, and Historical Consumption attribute sets.[270] 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.[271]

D.3.3.2 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[272] 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.

D.3.3.3 Metering Data Updates

The frequency and timeliness of updating metering data contained in the Metering[273] 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.[274]

D.3.3.3.1 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

269. CCB 940
270. CCB 1015
271. Incremental Release 1
272. CCB 940
273. CCB 940
274. Incremental Release 1

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*.[275]

D.3.3.4 Mirroring[276]

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.

D.3.3.4.1 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 D.3.3.4.2 below). 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.[277] 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.[278] This process may need to be repeated in the case of a Trust Center swap-out (refer to sub-clause 5.4.2.2.3 for further information).[279]

D.3.3.4.2 Mirror Attributes

The mandatory *Basic*, *Metering*, and (where applicable) *Prepayment*[280] 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.[281] The ZCL Not Authorized return status shall be used to provide

275. Incremental Release 1
276. CCB 1018
277. CCB 1289
278. CCB 1452
279. CCB 1419
280. CCB1218

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.[282]

D.4 Price Cluster

D.4.1 Overview

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[283] from either the utilities or from regional energy providers. The ESI[284] 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 and is outlined in Annex B.

281.CCB 1217
282.CCB 1262
283.CCB 1072
284.CCB 1072

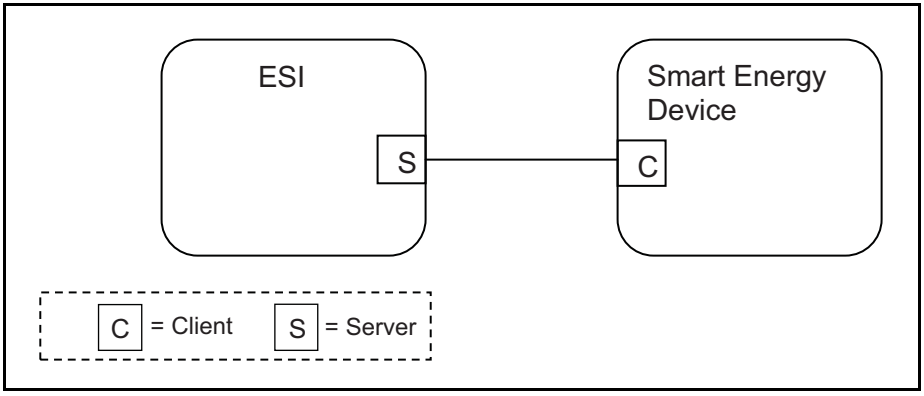

Figure D.18 Price Cluster Client Server Example

Please note the ESI[285] is defined as the Server due to its role in acting as the proxy for upstream price management systems and subsequent data stores.

D.4.2 Server

D.4.2.1 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 outlined in Annex B.

D.4.2.2 Attributes[286]

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 the following Table D.42.

Note: *The 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.*

285.CCB 1072
286.CCB4

Table D.42 Price Cluster Attribute Sets[a,b]

Attribute Set Identifier	Description
0x00	Tier Label
0x01	Block Threshold
0x02	Block Period
0x03	Commodity
0x04	Block Price Information
0x05	Reserved for future use
0x06	Reserved for future use
0x07	Billing Information Set
0x08 to 0xFF	Reserved

a. Incremental Release 1

b. CCB 1494

D.4.2.2.1 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 D.43 Tier Label Attribute Set[a]

Identifier	Name	Type	Length[b]	Access	Default	Mandatory / Optional
0x00[c]	*Tier1PriceLabel*	Octet String	1 to 13 Octets	Read/ Write	“Tier 1”	O
0x01[d]	*Tier2PriceLabel*	Octet String	1 to 13 Octets	Read/ Write	“Tier 2”	O
0x02[e]	*Tier3PriceLabel*	Octet String	1 to 13 Octets	Read/ Write	“Tier 3”	O
0x03[f]	*Tier4PriceLabel*	Octet String	1 to 13 Octets	Read/ Write	“Tier 4”	O
0x04[g]	*Tier5PriceLabel*	Octet String	1 to 13 Octets	Read/ Write	“Tier 5”	O
0x05[h]	*Tier6PriceLabel*	Octet String	1 to 13 Octets	Read/ Write	“Tier 6”	O
0x06[i]	*Tier7PriceLabel*	Octet String	1 to 13 Octets	Read only	“Tier 7”	O

Table D.43 Tier Label Attribute Set[a] (Continued)

Identifier	Name	Type	Length[b]	Access	Default	Mandatory / Optional
0x07[j]	*Tier8PriceLabel*	Octet String	1 to 13 Octets	Read only	“Tier 8”	O
0x08[k]	*Tier9PriceLabel*	Octet String	1 to 13 Octets	Read only	“Tier 9”	O
0x09[l]	*Tier10PriceLabel*	Octet String	1 to 13 Octets	Read only	“Tier 10”	O
0x0A[m]	*Tier11PriceLabel*	Octet String	1 to 13 Octets	Read only	“Tier 11”	O
0x0B[n]	*Tier12PriceLabel*	Octet String	1 to 13 Octets	Read only	“Tier 12”	O
0x0C[o]	*Tier13PriceLabel*	Octet String	1 to 13 Octets	Read only	“Tier 13”	O
0x0D[p]	*Tier14PriceLabel*	Octet String	1 to 13 Octets	Read only	“Tier 14”	O
0x0E[q]	*Tier15PriceLabel*	Octet String	1 to 13 Octets	Read only	“Tier 15”	O
0x0F to 0xFF[r]	Reserved					

a. Incremental Release 1
b. CCB 1292
c. Incremental Release 1
d. Incremental Release 1
e. Incremental Release 1
f. Incremental Release 1
g. Incremental Release 1
h. Incremental Release 1
i. Incremental Release 1
j. Incremental Release 1
k. Incremental Release 1
l. Incremental Release 1
m. Incremental Release 1
n. Incremental Release 1
o. Incremental Release 1
p. Incremental Release 1
q. Incremental Release 1
r. Incremental Release 1

D.4.2.2.1.1 *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”[287] and “Critical[288]”.

D.4.2.2.2 Block Threshold Set[289]

The following set of attributes 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 Pric*e command. The number of blocks is one greater than the number of thresholds.

Table D.44 Block Threshold Attribute Set[a]

Attribute Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0x00	*Block1Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x01	*Block2Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x02	*Block3Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x03	*Block4Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x04	*Block5Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O

287. CCB 1090
288. CCB 1090
289. Incremental Release 1

Table D.44 Block Threshold Attribute Set[a] (Continued)

Attribute Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0x05	*Block6Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x06	*Block7Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x07	*Block8Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x08	*Block9Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x09	*Block10Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x0A	*Block11Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x0B	*Block12Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x0C	*Block13Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x0D	*Block14Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x0E	*Block15Threshold*	Unsigned 48-bit Integer	0x000000000000 to 0xFFFFFFFFFFFF	Read Only	-	O
0x0F - 0xFF	Reserved					

a. Incremental Release 1

D.4.2.2.2.1 BlockNThreshold[290]

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.

D.4.2.2.3 Block Period Set[291]

The following set of attributes provides remote access to the Price server Block Threshold period (typically the billing cycle) information.

Table D.45 Block Period Attribute Set

Attribute Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x00	*StartofBlock Period*	UTCTime	-	Read Only	-	O
0x01	*BlockPeriod Duration (minutes)*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O
0x02	*ThresholdMultiplier*	Unsigned 24 bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O
0x03	*ThresholdDivisor*	Unsigned 24 bit Integer	0x000000 to 0xFFFFFF	Read Only	-	O
0x04 to 0xFF	Reserved					

290. Incremental Release 1
291. Incremental Release 1

D.4.2.2.3.1 *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[292], see sub-clause D.4.4.3 for further details.[293]

D.4.2.2.3.2 *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.[294]

D.4.2.2.3.3 *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 D.3.2.2.1.13). This attribute must be used in conjunction with the *ThresholdDivisor* attribute. An attribute value of zero shall result in a unitary multiplier (0x000001).[295]

D.4.2.2.3.4 *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 D.3.2.2.1.13). This attribute must be used in conjunction with the *ThresholdMultiplier* attribute. An attribute value of zero shall result in a unitary divisor (0x000001).[296]

D.4.2.2.4 Commodity Set[297]

The following set of attributes 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.*

292. CCB 1537
293. CCB 1547
294. CCB 1537
295. CCB 1300
296. CCB 1300
297. Incremental Release 1

This feature set may change before reaching certifiable status in a future revision of this specification.

Table D.46 *Commodity* Attribute Set

Attribute Identifier [a]	Name	Type	Range	Access	Default	Mandatory / Optional
0x00	*CommodityType*	8-bit Enumeration	0x00 to 0xFF	Read Only	-	O
0x01	*Standing Charge*	Unsigned 32 bit Integer	0x00000000 to 0xFFFFFFFF	Read Only	-	O
0x02	*ConversionFactor*	Unsigned 32 bit Integer	0x00000000 to 0xFFFFFFFF	Read Only	0x10000000	O
0x03	*ConversionFactorTrailingDigit*	8-bit BitMap		Read Only	0x70	O
0x04	*CalorificValue*	Unsigned 32 bit Integer	0x00000000 to 0xFFFFFFFF	Read Only	0x2625A00	O
0x05	*CalorificValueUnit*	8-bit Enumeration		Read Only	0x1	O
0x06	*CalorificValueTrailingDigit*	8-bit BitMap		Read Only	0x60	O
0x07 - 0xFF	Reserved					

a. CCB 1264

D.4.2.2.4.1 *CommodityType* Attribute[298]

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 D.23).

298. Incremental Release 1

D.4.2.2.4.2 *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 D.4.2.4.1). A value of 0xFFFFFFFF indicates field not used.[299]

D.4.2.2.4.3 *ConversionFactor* Attribute[300]

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.

D.4.2.2.4.4 *ConversionFactorTrailingDigit* Attribute[301]

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.

D.4.2.2.4.5 *CalorificValue* Attribute[302]

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.

D.4.2.2.4.6 *CalorificValueUnit* Attribute[303]

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 D.47 below. The *CalorificValueUnit* attribute represents the current active value.

299. Incremental Release 1
300. CCB 1264
301. CCB 1264
302. CCB 1264
303. CCB 1264

Table D.47 Values and Descriptions for the *CalorificValueUnit* Attribute

Values	Description
0x00	Reserved for future use
0x01	MJ/m3
0x02	MJ/kg
0x03 to 0xFF	Reserved for future use

D.4.2.2.4.7 *CalorificValueTrailingDigit* Attribute[304]

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.

D.4.2.2.5 Block Price Information Set[305]

The following set of attributes 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 D.48 Block Price Information Attribute Set

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x00	*NoTierBlock1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x01	*NoTierBlock2Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x02	*NoTierBlock3Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O

304. CCB 1264
305. Incremental Release 1

Table D.48 Block Price Information Attribute Set (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x0N	*NoTierBlockN+1Price …*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x0F	*NoTierBlock16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x10	*Tier1Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x11	*Tier1Block2Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x12	*Tier1Block3Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x1N	*Tier1BlockN+1Price …*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x1F	*Tier1Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x20	*Tier2Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x2N	*Tier2BlockN+1Price …*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x2F	*Tier2Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O

Table D.48 Block Price Information Attribute Set (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x30	*Tier3Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x3N	*Tier3BlockN+1Price …*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x3F	*Tier3Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x40	*Tier4Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x4N	*Tier4BlockN+1Price …*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x4F	*Tier4Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x50	*Tier5Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x5N	*Tier5BlockN+1Price …*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x5F	*Tier5Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x60	*Tier6Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O

Table D.48 Block Price Information Attribute Set (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x6N	*Tier6BlockN+1Price …*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x6F	*Tier6Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x70	*Tier7Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x7N	*Tier7BlockN+1Price …*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x7F	*Tier7Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x80	*Tier8Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x8N	*Tier8BlockN+1Price …*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x8F	*Tier8Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x90	*Tier9Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x9N	*Tier9BlockN+1Price …*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O

Table D.48 Block Price Information Attribute Set (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x9F	*Tier9Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xA0	*Tier10Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xAN	*Tier10BlockN+1Price* …	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xAF	*Tier10Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xB0	*Tier11Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xBN	*Tier11BlockN+1Price* …	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xBF	*Tier11Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xC0	*Tier12Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xCN	*Tier12BlockN+1Price* …	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xCF	*Tier12Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O

Table D.48 Block Price Information Attribute Set (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0xD0	*Tier13Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xDN	*Tier13BlockN+1Price* …	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xDF	*Tier13Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xE0	*Tier14Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xEN	*Tier14BlockN+1Price* …	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xEF	*Tier14Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xF0	*Tier15Block1Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xFN	*Tier15BlockN+1Price* …	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O
0xFF	*Tier15Block16Price*	Unsigned 32-bit integer	0x00000000 to 0xFFFFFFFF	Read only	-	O

D.4.2.2.5.1 *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 D.4.2.4.1). 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.[306]

D.4.2.2.6 Billing Information Attribute Set[307]

The following set of attributes provides remote access to the Price server Billing information.

Table D.49 Billing Information Attribute Set

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x00	*CurrentBillingPeriodStart*	UTCTime	0x00000000 to 0xFFFFFFFF	Read only	-	O
0x01	*CurrentBillingPeriodDuration*	Unsigned 24-bit Integer	0x000000 to 0xFFFFFF	Read only	-	O

D.4.2.2.6.1 *CurrentBillingPeriodStart* Attribute

The *CurrentBillingPeriodStart* attribute represents the start time of the current billing period.

D.4.2.2.6.2 *CurrentBillingPeriodDuration* Attribute

The *CurrentBillingPeriodDuration* attribute represents the current billing period duration in minutes.[308]

D.4.2.3 Commands Received

The server side of the Price cluster is capable of receiving the commands listed in Table D.50.

306. CCB 1547
307. CCB 1494
308. CCB 1494

Table D.50 Received Command IDs for the Price Cluster

Command Identifier Field Value[a]	Description	Mandatory / Optional
0x00	*Get Current Price*	M
0x01	*Get Scheduled Prices*	O
0x02[b]	*Price Acknowledgement*[c]	M - Mandatory for 1.1 and later devices[d]
0x03[e]	*Get Block Period(s)*	O
0x04	*GetConversionFactor*	O
0x05	*GetCalorificValue*	O
0x06 to 0xFF[f]	Reserved	

a. CCB 1264

b. Incremental Release 1

c. Incremental Release 1

d. CCB 1207

e. Incremental Release 1

f. Incremental Release 1

D.4.2.3.1 *Get Current Price* Command

This command initiates a *Publish Price* command (see sub-clause D.4.2.4.1) for the current time.

D.4.2.3.1.1 Payload Format

The payload of the *Get Current Price* command is formatted as shown in Figure D.19:

Octets	1
Data Type	Unsigned 8-bit integer
Field Name	Command Options

Figure D.19 The Format of the *Get Current Price* Command Payload

D.4.2.3.1.1.1 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 D.20.

Bits	0	1 to 7
Field Name	Requestor Rx On When Idle	Reserved

Figure D.20 *Get Current Price* Command Options Field

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.

D.4.2.3.1.2 Effect on Receipt

On receipt of this command, the device shall send a *Publish Price* command (sub-clause D.4.2.4.1) 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[309] shall respond on the Inter-Pan. If the *GetCurrentPrice* command is received on the SE Network, the ESI[310] shall respond to the device requesting the pricing information.

D.4.2.3.2 *Get Scheduled Prices* Command

This command initiates a *Publish Price* command (see sub-clause D.4.2.4.1) for [311]available price events.[312] A server device shall be capable of storing five [313]price events at a minimum.

309. CCB 1072
310. CCB 1072
311. CCB 1294
312. CCB 1087
313. CCB 1087

D.4.2.3.2.1 Payload Details

The *Get Scheduled Prices* command payload shall be formatted as illustrated in Figure D.21:

Octets	**4**	**1**
Data Type	UTCTime	Unsigned8-bit integer
Field Name	Start Time (M)	Number of Events (M)

Figure D.21 Format of the *Get Scheduled Prices* Command Payload

Start Time (mandatory): UTC Timestamp representing the minimum ending time for any scheduled or currently active pricing events to be resent. If a[314]command has a Start Time of 0x00000000, replace that Start Time with the current time stamp[315].

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).

D.4.2.3.2.2 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.[316]

A ZCL Default Response with status NOT_FOUND shall be returned if there are no events available.[317]

D.4.2.3.2.3 Effect on Receipt

On receipt of this command, the device shall send a *Publish Price* command (see sub-clause D.4.2.4.1) 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[318] shall respond on the Inter-Pan. If the *GetScheduledPrices*

314.CCB 1294
315.CCB 1244
316.CCB 1087
317.CCB 1119

command is received on the SE Network, the ESI[319] shall respond to the device requesting the pricing information.

D.4.2.3.3 *Price Acknowledgement* Command[320]

The *Price Acknowledgement* command described in Figure D.22 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.[321]

D.4.2.3.3.1 Payload Format

Octets	4	4	4	1
Data Type	Unsigned 32 bit Integer	Unsigned 32 bit Integer	UTCTime	8 bit BitMap
Field Name	Provider ID (M)	Issuer Event ID (M)	Price Ack Time (M)	Control (M)

Figure D.22 Format of the *Price Acknowledgement* Command Payload

D.4.2.3.3.1.1 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 D.26 and Figure D.27.

D.4.2.3.3.2 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 D.4.2.4.1 for further details).[322]

318. CCB 1072
319. CCB 1072
320. Incremental Release 1
321. CCB 1207
322. CCB 1206

D.4.2.3.4 *Get Block Period(s)* Command[323]

This command initiates a *Publish Block Period* command (see sub-clause D.4.2.4.2) 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 may change before reaching certifiable status in a future revision of this specification.*

D.4.2.3.4.1 Payload Format

Octets	4	1
Data Type	UTCTime	Unsigned 8 bit Integer
Field Name	Start Time (M)	Number of Events (M)

Figure D.23 Format of the *Get Block Period(s)* Command Payload

D.4.2.3.4.1.1 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[324] command has a Start Time of 0x00000000, replace that Start Time with the current time stamp.[325]

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.

D.4.2.3.4.2 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.

323. Incremental Release 1
324. CCB 1294
325. CCB 1244

A ZCL Default response with status NOT_FOUND shall be returned if there are no events available.

D.4.2.3.4.3 Effect on Receipt

On receipt of this command, the device shall send a *Publish Block Period* command (sub-clause D.4.2.4.2) for all currently scheduled periods, up to the maximum number of commands specified.

D.4.2.3.5 *GetConversionFactor* Command[326]

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 may change before reaching certifiable status in a future revision of this specification.*

D.4.2.3.5.1 Payload Format

Octets	4	4
Data Type	UTC Time	Unsigned 8-bit Integer
Field Name	Start Time	Number of Events

Figure D.24 Format of the *GetConversionFactor* Command Payload

D.4.2.3.5.2 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

326.CCB 1264

one instance is requested, the active and scheduled instances shall be sent with ascending ordered StartTime.

D.4.2.3.6 *GetCalorificValue* Command[327]

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 may change before reaching certifiable status in a future revision of this specification.*

D.4.2.3.6.1 Payload Format

Octets	4	1
Data Type	UTC Time	Unsigned 8-bit Integer
Field Name	Start Time	Number of Events

Figure D.25 Format of the *GetCalorificValue* Command Payload

D.4.2.3.6.2 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.

D.4.2.4 Commands Generated

The server side of the Price cluster is capable of generating the commands listed in Table D.51.

327.CCB 1264

Table D.51 Generated Command IDs for the Price Cluster

Command Identifier Field Value	Description	Mandatory / Optional
0x00	*Publish Price*	M
0x01[a]	*Publish Block Period*	O
0x02[b]	*Publish Conversion Factor*	O
0x03[c]	*Publish Calorific Value*	O
0x04[d] – 0xFF[e]	Reserved	

a. Incremental Release 1

b. CCB 1264

c. CCB 1264

d. CCB 1264

e. Incremental Release 1

D.4.2.4.1 *Publish Price* Command

The *Publish Price* command is generated in response to receiving a *Get Current Price* command (see sub-clause D.4.2.3.1), in response to a *Get Scheduled Prices* command (see sub-clause D.4.2.3.2), and when an update to the pricing information is available from the commodity provider, either before, or when a TOU price becomes active.[328, 329, 330] Additionally the *Publish Price* command is generated as specified in sub-clause D.4.4.3 when Block Pricing is in effect.[331]

When a *Get Current Price* or *Get Scheduled Prices* command is received over a ZigBee 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 ZigBee 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

328. CCB 1537
329. CCB 1347
330. CCB 1083
331. CCB 1547

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.[332] [333]

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.[334] 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.[335]

D.4.2.4.1.1 Payload Format

The *PublishPrice* command payload shall be formatted as illustrated in Figure D.26.

332. CCB 1083
333. CCB 1389 (deleted text)
334. CCB 1072
335. CCB 1083

Octets	4	1-13[a]	4	4	1	2	1
Data Type	Unsigned 32-bit Integer	Octet String	Unsigned 32-bit Integer	UTCTime	8 bits enumeration	Unsigned 16-bit Integer	8-bit BitMap
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	8-bit BitMap	UTCTime	Unsigned 16-bit Integer	Unsigned 32-bit Integer	Unsigned 8-bit Integer	Unsigned 32-bit Integer	Unsigned 8-bit Integer
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	Unsigned 32-bit Integer	8-bit enumeration	8-bit BitMap	8 bit Integer	8-bit BitMap
Field Name	Alternate Cost Delivered (O)[d]	Alternate Cost Unit (O)[e]	Alternate Cost Trailing Digit(O)[f]	Number of Block Thresholds (O)	Price Control (O)

a. CCB 1292

b. Incremental Release 1

c. Incremental Release 1

d. CCB 973

e. CCB 973

f. CCB 973

Figure D.26 Format of the *Publish Price* Command Payload

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.*[336]

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 (UTCTime 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.[337]

Current Time (mandatory): A UTCTime 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[338] cluster (see sub-clause D.3.2.2.4.1).

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.[339] This sub-field also references the associated *TiernPriceLabel* attribute assigned to the Price Tier. Table D.52 depicts the assignments:

336. CCB 1070
337. CCB 1389
338. CCB 940
339. CCB1 1332

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 D.52 Price Tier Sub-field Enumerations

Enumerated Value[a]	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*

a. CCB 1268

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 D.3.2.2.1.29, 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 D.53.

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 D.53 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 UTCTime field to denote the time at which the price signal becomes valid. A Start Time of 0x00000000 is a special time denoting "now."[340]

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.[341]

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 D.4.4.3 for further details[342]), the Duration in Minutes field of the

340.CCB 1002
341.CCB 1243
342.CCB 1547

Publish Price command shall be set to 0xFFFF indicating the price is valid “until changed”.[343]

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.[344]

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.[345]

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 D.54) for the Alternate Cost Delivered field. A value of 0xFF indicates the field is not used.[346]

343. CCB 1537
344. CCB 1333
345. CCB 1333
346. CCB 1210

Table D.54 Alternate Cost Unit Enumerations

Values	Description
0x00	Reserved for future use
0x01	Kg of CO_2 per unit of measure
0x02 to 0xFF	Reserved for future use

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.[347] A value of 0xFF indicates the field is not used.[348]

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[349], see sub-clause D.4.4.3 for further details.[350]

Price Control (optional): Identifies additional control options for the price event. A value of 0x00 indicates field not used. Note that for ZigBee SE 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.[351]

The BitMap for this field is described in Table D.55.[352]

Table D.55 Price Control Field BitMap[a]

Bit	Description
0	1=Price Acknowledgement required, 0=Price Acknowledgement not required
1 to 7	Reserved

a. Incremental Release 1

347. CCB 973
348. CCB 1210
349. CCB 1537
350. CCB 1547
351. CCB 1207
352. Incremental Release 1

D.4.2.4.1.2 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.[353,354]

D.4.2.4.2 *Publish Block Period* Command[355]

The *Publish Block Period* command is generated in response to receiving a *Get Block Period(s)* command (see sub-clause D.4.2.3.4) 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 ZigBee 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 ZigBee 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.*

353. CCB CC-900
354. CCB 1294
355. Incremental Release 1

D.4.2.4.2.1 Payload Format

Octets	4	4	4	3	1	1
Data Type	Unsigned 32 bit Integer	Unsigned 32 bit Integer	UTCTime	Unsigned 24 bit Integer	8 bit BitMap	8 bit BitMap
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)

Figure D.27 Format of the *Publish Block Period* Command Payload

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 (UTCTime 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 UTCTime 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.[356]

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

356.CCB 1243

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 D.56.

Table D.56 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[a]
2-7	Reserved

a. CCB 1294

Repeating Block: Indicates whether a block period repeats on expiry.

D.4.2.4.3 *PublishConversionFactor* Command[357]

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.*

357.CCB 1264

D.4.2.4.3.1 Payload Format

Octets	4	4	4	1
Data Type	Unsigned 32-bit Integer	UTC Time	Unsigned 32-bit Integer	8-bit BitMap
Field Name	Issuer Event ID (M)	Start Time (M)	Conversion Factor (M)	Conversion Factor Trailing Digit (M)

Figure D.28 Format of the *PublishConversionFactor* Command Payload

D.4.2.4.3.2 Payload Details

Issuer Event ID (mandatory): Unique identifier generated by the commodity provider.

Start Time (mandatory): A UTCTime 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 D.4.2.2.4.3).

Conversion Factor Trailing Digit (mandatory): See Price Cluster Commodity attributes (see sub-clause D.4.2.2.4.4).

D.4.2.4.4 *PublishCalorificValue* Command[358]

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.*

358.CCB 1264

D.4.2.4.4.1 Payload Format

Octets	4	4	4	1	1
Data Type	Unsigned 32-bit Integer	UTC Time	Unsigned 32-bit Integer	8-bit Enumeration	8-bit BitMap
Field Name	Issuer Event ID (M)	Start Time (M)	Calorific Value (M)	Calorific Value Unit (M)	Calorific Value Trailing Digit (M)

Figure D.29 Format of the *PublishCalorificValue* Command Payload

D.4.2.4.4.2 Payload Details

Issuer Event ID (mandatory): Unique identifier generated by the commodity provider.

Start Time (mandatory): A UTCTime 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 D.4.2.2.4.5).

Calorific Value Unit (mandatory): See Price Cluster Commodity attributes (see sub-clause D.4.2.2.4.6).

Calorific Value Trailing Digit (mandatory): See Price Cluster Commodity attributes (see sub-clause D.4.2.2.4.7).

D.4.3 Client

D.4.3.1 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 outlined in Annex B.

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.*

D.4.3.2 Attributes[359]

Table D.57 Price Client Cluster Attributes

Attribute Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0x0000	*PriceIncrease Randomize Minutes*	Unsigned 8-bit Integer	0x00 to 0x3C	Read/ Write	0x05	O
0x0001	*PriceDecrease Randomize Minutes*	Unsigned 8-bit Integer	0x00 to 0x3C	Read/ Write	0x0F	O
0x0002	*Commodity Type*	8-bit Enumeration	0x00 to 0xFF	Read Only	-	O
0x0003 - 0x000F	Reserved					

D.4.3.2.1 *PriceIncreaseRandomizeMinutes* Attribute[360]

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.[361]

359. Incremental Release 1
360. Incremental Release 1
361. CCB 1293

D.4.3.2.2 *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[362] 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.[363]

D.4.3.2.3 *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 D.23).[364]

D.4.3.3 Commands Received

The client receives the cluster-specific response commands detailed in sub-clause D.4.2.4.[365]

D.4.3.4 Commands Generated

The client generates the cluster-specific commands detailed in sub-clause D.4.2.3, as required by the application.[366]

362. CCB 1294
363. Incremental Release 1
364. Incremental Release 1
365. CCB 1031
366. CCB 1031

D.4.4 Application Guidelines[367]

D.4.4.1 Registering for Commands

Devices should use bind request to register for unsolicited *Publish*[368] *Price*, *Display Message* and *Load Control Event* commands.

D.4.4.2 Attribute Reporting[369]

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 D.3.3.2. 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.[370]

D.4.4.3 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.[371]

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.[372]

D.4.4.3.1 TOU Charging Only[373]

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.[374]

D.4.4.3.2 Block Charging only[375]

Indicated[376] by the *Number of Price Tiers* fields being set to zero while the *Number of Block Thresholds* is between 0x01 and 0x0F.

367. CCB 1032
368. CCB 1334
369. Incremental Release 1
370. Incremental Release 1
371. CCB 1547
372. CCB 1547
373. Incremental Release 1
374. CCB 1547 (text moved)
375. Incremental Release 1

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.[377]

D.4.4.3.3 Block/TOU Combination Charging[378]

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.*[379]

376. CCB 1294
377. CCB 1547
378. Incremental Release 1
379. CCB 1547

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.

D.4.4.3.4 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.[380]

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.[381]

380. Incremental Release 1
381. Incremental Release 1

D.5 Messaging Cluster

D.5.1 Overview

This cluster provides an interface for passing text messages between ZigBee devices. Messages are expected to be delivered via the ESI[382] and then unicast to all individually registered devices implementing the Messaging Cluster on the ZigBee 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[383].

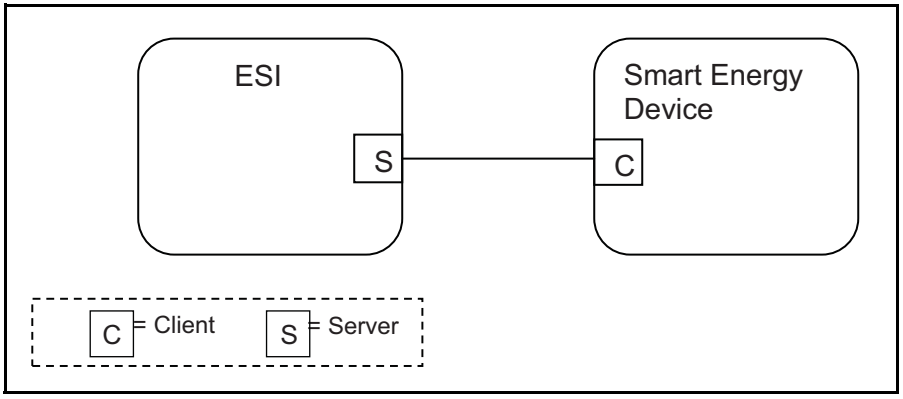

Figure D.30 Messaging Cluster Client/Server Example

Please note the ESI[384] is defined as the Server due to its role in acting as the proxy for upstream message management systems and subsequent data stores.

D.5.2 Server

D.5.2.1 Dependencies

Support for ZCL Data Types.

Anonymous Inter-PAN transmission mechanism outlined in Annex B.

No dependencies exist for other Smart Energy Clusters.

382.CCB 1072
383.CCB 1072
384.CCB 1072

D.5.2.2 Attributes

None.

D.5.2.3 Commands Generated

The command IDs generated by the Messaging server cluster are listed in Table D.58.

Table D.58 Generated Command IDs for the Messaging Server

Command Identifier Field Value	Description	Mandatory / Optional
0x00	*Display Message*	M
0x01	*Cancel Message*	M
0x02 – 0xff	Reserved	

D.5.2.3.1 *Display Message* Command

D.5.2.3.1.1 Payload Format

The *Display Message* command payload shall be formatted as illustrated in Figure D.31.

Octets	4	1	4	2	Variable
Data Type	Unsigned 32-bit integer	8-bit BitMap	UTCTime	Unsigned 16-bit Integer	Character string
Field Name	Message ID	Message Control	Start Time	Duration In Minutes	Message

Figure D.31 Format of the *Display Message* Command Payload

D.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 (UTCTime 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 outlined in Annex B or that a user confirmation is required for a message. Bit encoding of this field is outlined in Table D.59:

Table D.59 Message Control Field Bit Map

Bits	Enumeration	Value	Description
Bits 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.
	Anonymous Inter-PAN transmission only	2	Send message through the Anonymous Inter-PAN transmission mechanism.
	Reserved	3	Reserved value for future use.
Bits 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.
Bits 4 to 6	Reserved	N/A	These bits are reserved for future use.
Bit 7	Message Confirmation	0	Message Confirmation not required.
		1	Message Confirmation required.

If the Anonymous Inter-PAN transmission mechanism outlined in Annex B 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.[385] Confirmation is typically used when the Utility is sending down information such as a disconnection notice, or prepaid billing information.[386]

385. CCB 1276
386. CCB 996

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 UTCTime 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. [387]

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 outlined in Annex B 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.[388] 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 sub-clause 5.3.8 for further details on Fragmentation settings.[389]

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.[390]

D.5.2.3.2 *Cancel Message* Command

The *Cancel Message* command described in Figure D.32 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.

387. CCB 1243
388. CCB 1027
389. CCB 1027
390. CCB 1027

Octets	4	1
Data Type	Unsigned 32-bit integer	8-bit BitMap
Field Name	Message ID	Message Control

Figure D.32 Format of the *Cancel Message* Command Payload

D.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 (UTCTime 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 outlined in Annex B. 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 D.59.

D.5.3 Client

D.5.3.1 Dependencies

Support for ZCL Data Types.

No dependencies exist for other Smart Energy Clusters.

D.5.3.2 Attributes

None.

D.5.3.3 Commands Generated

The command IDs generated by the Messaging cluster are listed in Table D.60.

Table D.60 Messaging Client Commands

Command Identifier Field Value	Description	Mandatory / Optional
0x00	*Get Last Message*	M
0x01	*Message Confirmation*	M
0x02 – 0xff	Reserved	

D.5.3.3.1 *Get Last Message* Command

This command has no payload.

D.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 D.5.2.3.1). A ZCL Default Response with status NOT_FOUND shall be returned if no message is available.[391]

D.5.3.3.2 *Message Confirmation* Command

The *Message Confirmation* command described in Figure D.33 provides the ability to acknowledge a previously sent message.

Octets	4	4
Data Type	Unsigned 32-bit integer	UTCTime
Field Name	Message ID	Confirmation Time

Figure D.33 Format of the *Message Confirmation* Command Payload

D.5.3.3.2.1 Payload Details

Message ID: A unique unsigned 32-bit number identifier for the message being confirmed.[392]

Confirmation Time: UTCTime of user confirmation of message.

D.5.4 Application Guidelines

For Server and Client transactions, please refer to Figure D.34.[393]

391.CCB 1119
392.CCB 1096
393.CCB 1294

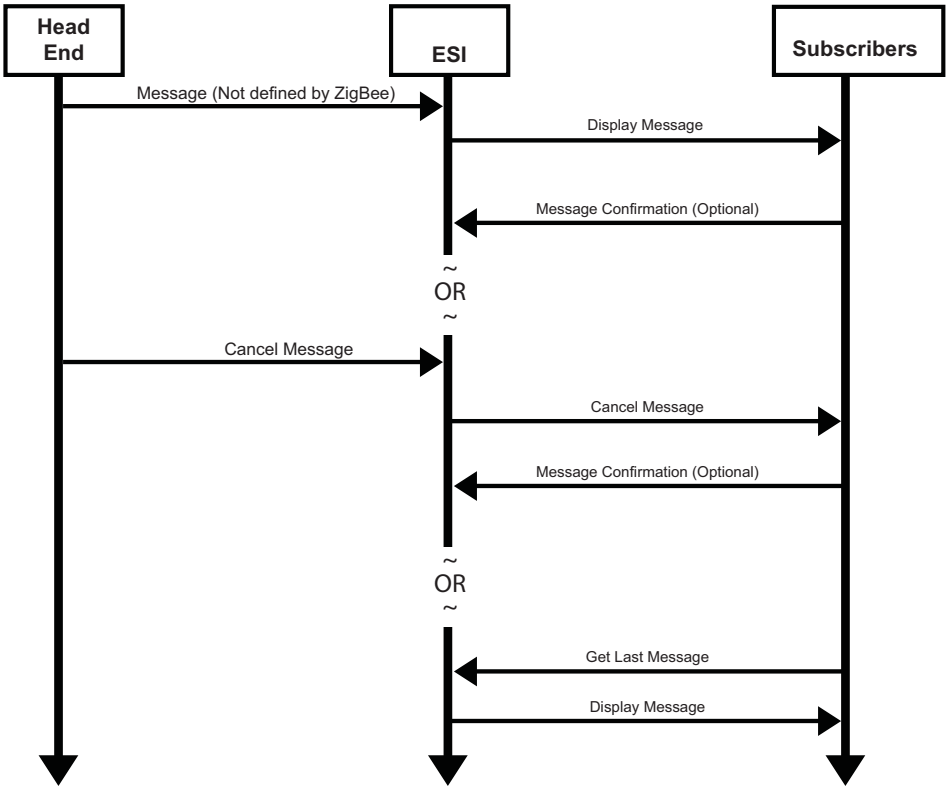

Figure D.34 Client/Server Message Command Exchanges

D.6 Tunneling Cluster[394]

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.*

D.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).

394. Incremental Release 1

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.

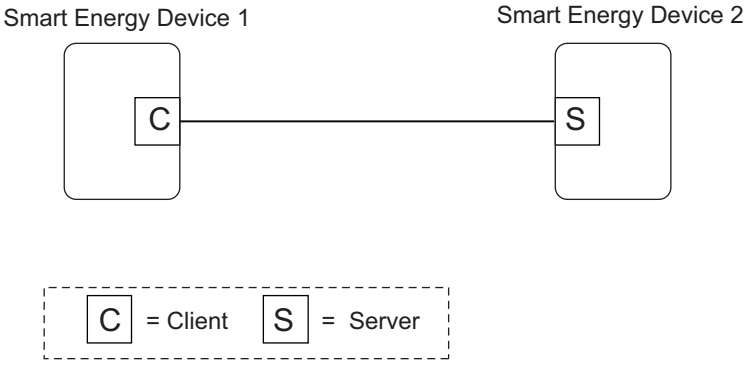

Figure D.35 A Client Requests a Tunnel From a Server to Exchange Complex Data in Both Directions

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 read/write 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 (T*ransferDataError*).

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[395] (*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.[396] 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 (Figure D.36) and with (Figure D.37) flow control.

395.CCB 1353
396.CCB 1353

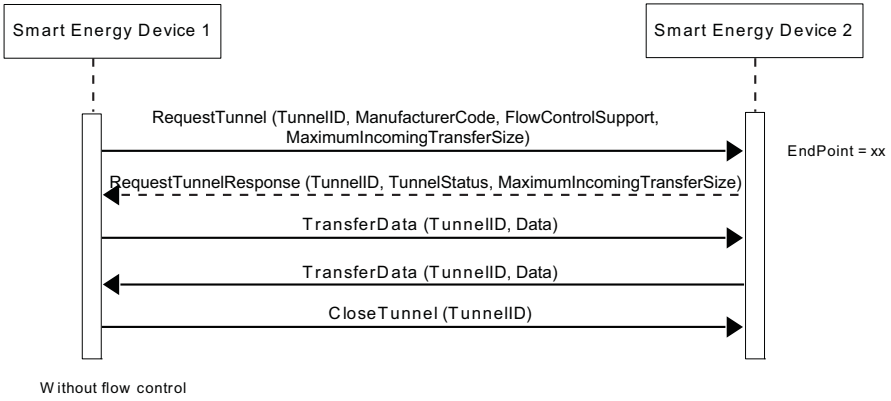

Figure D.36 SE Device 1 (Client) Requests a Tunnel From SE Device 2 (Server) to Transfer Data Without Flow Control (Default)

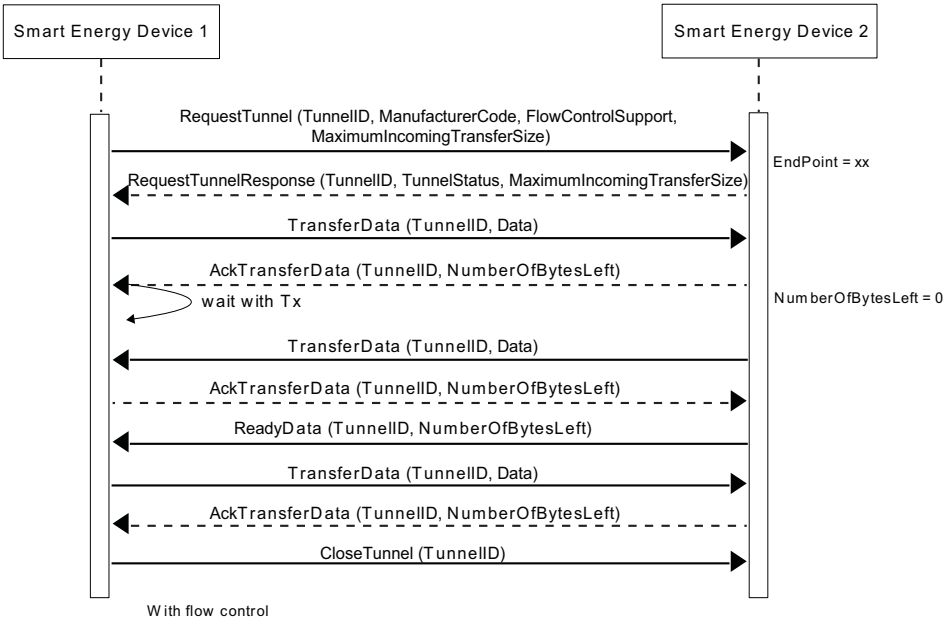

Figure D.37 SE Device 1 (Client) Requests a Tunnel From SE Device 2 (Server) to Transfer Data With Flow Control

D.6.2 Server

D.6.2.1 Dependencies

This cluster requires APS fragmentation [B3] to be implemented, with maximum transfer sizes defined by the device's negotiated input buffer sizes.[397]

D.6.2.2 Attributes

Table D.61 Tunneling Cluster Attributes

Identifier	Name	Type	Range	Access	Default	Man. /Opt.
0x00	*CloseTunnelTimeout*	Unsigned 16-bit Integer	0x0001[a]-0xFFFF	Read Only	0xFFFF	M

a. CCB 1355

D.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.[398]

397.CCB 1353
398.CCB 1355

D.6.2.3 Parameters

The table below 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 D.62 Cluster Parameters Passed Through Commands

Name	Type	Range	Default	Mandatory / Optional
ProtocolID	8-bit enumeration	0x01 – 0xFF	0x00	M
ManufacturerCode	Unsigned 16-bit integer	0x0000 – 0xFFFF	0x00	M
FlowControlSupport	Boolean	TRUE or FALSE	FALSE	M
MaximumIncomingTransferSize [a]	Unsigned 16-bit integer	0x0000 – 0xFFFF	1500	M
TunnelID	Unsigned 16-bit integer	0x0000 – 0xFFFF	(Return value)	M
Data	Octet string	-	-	M
NumberOfOctetsLeft	Unsigned 16-bit integer	0x0000 – 0xFFFF	-	M
TunnelStatus	Unsigned 8-bit integer	0x00 – 0x04	-	M
TransferDataStatus	Unsigned 8-bit integer	0x00 – 0x01	-	M

a. CCB 1353

D.6.2.4 Commands Received

Table D.63 lists cluster-specific commands received by the server.

Table D.63 Cluster -specific Commands Received by the Server

Command Identifier FieldValue	Description	Mandatory/ Optional
0x00	*RequestTunnel*	M
0x01	*CloseTunnel*	M
0x02	*TransferData*	M
0x03	*TransferDataError*	M
0x04	*AckTransferData*	O
0x05	*ReadyData*	O
0x06[a]	*GetSupportedTunnelProtocols*	O

a. CCB 1273

D.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.

D.6.2.4.1.1 Payload Format

Octets	1	2	1	2
Data Type	8-bit enumeration	Unsigned 16-bit integer	Boolean	Unsigned 16-bit integer
Field Name	ProtocolID (M)	Manufacturer Code (M)	FlowControl Support (M)	Maximum Incoming TransferSize[a]

a. CCB 1353

Figure D.38 Format of the *RequestTunnel* Command Payload

D.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 D.64 lists the

possible values for the *ProtocolID*. The values above 199 may be used for manufacturer specific protocols.

Table D.64 ProtocolID Enumerations

Values	Description
0	DLMS/COSEM (IEC 62056)
1	IEC 61107
2	ANSI C12
3	M-BUS
4	SML
5	ClimateTalk
6 to 199	Reserved for future growth
200 to 254	Manufacturer-defined protocols
255	Reserved

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.

D.6.2.4.1.3 When Generated

Is never generated by the server.

D.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.

D.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.

D.6.2.4.2.1 Payload Format

Octets	2
Data Type	Unsigned 16-bit integer
Field Name	TunnelID (M)

Figure D.39 Format of the *CloseTunnel* Command Payload

D.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.

D.6.2.4.2.3 When Generated

This command is never generated by the server.

D.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.

D.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.

D.6.2.4.3.1 Payload Format

Octets	2	Variable
Data Type	Unsigned 16-bit integer	Octets
Field Name	TunnelID (M)	Data (M)

Figure D.40 Format of the *TransferData* Command Payload

D.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: Octet 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.

D.6.2.4.3.3 When Generated

Is generated whenever the server wants to tunnel protocol data to the client.

D.6.2.4.3.4 Effect on Receipt

Indicates that the server has received tunneled protocol data from the client.

D.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.

D.6.2.4.4.1 Payload Format

Octets	2	1
Data Type	Unsigned 16-bit integer	Unsigned 8-bit integer
Field Name	TunnelID (M)	TransferDataStatus (M)

Figure D.41 Format of the *TransferDataError* Command Payload

D.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 D.65.

Table D.65 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
0x03 – 0xFF	Reserved	Should not be returned and indicates an unknown error.

D.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.

D.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.

D.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.

D.6.2.4.5.1 Payload Format

Octets	2	2
Data Type	Unsigned 16-bit integer	Unsigned 16-bit Integer
Field Name	TunnelID (M)	NumberOfBytes Left (M)

Figure D.42 Format of the *AckTransferData* Command Payload

D.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.

D.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.

D.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.

D.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.

D.6.2.4.6.1 Payload Format

Octets	2	2
Data Type	Unsigned 16-bit integer	Unsigned 16-bit Integer
Field Name	TunnelID (M)	NumberOfOctets Left (M)

Figure D.43 Format of the *ReadyData* Command Payload

D.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.

D.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.

D.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.

D.6.2.4.7 *Get Supported Tunnel Protocols* Command[399]

Get Supported Tunnel Protocols is the client command used to determine the tunnel protocols supported on another device.

D.6.2.4.7.1 Payload Format

Octets	1
Data Type	Unsigned 8-bit Integer
Field Name	Protocol Offset

Figure D.44 Format of the *Get Supported Tunnel Protocols* Command Payload

D.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.

D.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 D.6.2.5.6 for further details).

D.6.2.5 Commands Generated

Table D.66 lists commands that are generated by the server.

399.CCB 1273

Table D.66 Cluster-Specific Commands Sent by the Server

Command Identifier FieldValue	Description	Mandatory/ Optional
0x00	*RequestTunnelResponse*	M
0x01	*TransferData*	M
0x02	*TransferDataError*	M
0x03	*AckTransferData*	O
0x04	*ReadyData*	O
0x05[a]	*Supported Tunnel Protocols Response*	O
0x06[b]	*TunnelClosureNotification*	O

a. CCB 1273

b. CCB 1401

D.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.

D.6.2.5.1.1 Payload Format

Octets	2	1	2
Data Type	Unsigned 16-bit Integer	Unsigned 8-bit Integer	Unsigned 16-bit Integer
Field Name	TunnelID (M)	TunnelStatus (M)	Maximum Incoming TransferSize[a]

a. CCB 1353

Figure D.45 Format of the *RequestTunnelResponse* Command Payload

D.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 D.67.

Table D.67 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).
0x05 to 0xFF	Reserved	Should not be returned and indicates an unknown error.

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.

D.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.

D.6.2.5.1.4 Effect on Receipt

Should never be received by the server.

D.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.

D.6.2.5.2.1 Payload Format

Octets	2	Variable
Data Type	Unsigned 16-bit integer	Octets
Field Name	TunnelID (M)	Data (M)

Figure D.46 Format of the *TransferData* Command Payload

D.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: 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).

D.6.2.5.2.3 When Generated

Is generated when the server wants to tunnel protocol data to the client.

D.6.2.5.2.4 Effect on Receipt

Indicates that the server has received tunneled protocol data from the client.

D.6.2.5.3 *TransferDataError* Command

See sub-clause D.6.2.4.4.

D.6.2.5.4 *AckTransferData* Command

See sub-clause D.6.2.4.5.

D.6.2.5.5 *ReadyData* Command

See sub-clause D.6.2.4.6.

D.6.2.5.6 *Supported Tunnel Protocols Response* Command[400]

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.

D.6.2.5.6.1 Payload Format

Octets	1	1	3	…	3
Data Type	Boolean	Unsigned 8-bit Integer			
Field Name	Protocol List Complete	Protocol Count	Protocol 1	…	Protocol n

Figure D.47 Format of the *Supported Tunnel Protocols Response* Command Payload

where each protocol field shall be formatted as:

Octets	2	1
Data Type	Unsigned 16-bit Integer	8-bit Enumeration
Field Name	Manufacturer Code	Protocol ID

Figure D.48 Format of the *Supported Tunnel Protocols Response* Command Protocol Fields

D.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

400.CCB 1273

Protocol ID: An enumeration representing the identifier of the metering communication protocol for the supported tunnel. Table D.64 lists the possible values for standard protocols

D.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

D.6.2.5.7 *TunnelClosureNotification* Command[401]

TunnelClosureNotification is sent by the server to indicate that a tunnel has been closed due to expiration of a *CloseTunnelTimeout*.

D.6.2.5.7.1 Payload Format

Octets	2
Data Type	Unsigned 16-bit Integer
Field Name	TunnelID (M)

Figure D.49 Format of the *TunnelClosureNotification* Command Payload

D.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.

D.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.

D.6.3 Client

D.6.3.1 Dependencies

This cluster requires APS fragmentation [B3] to be implemented, with maximum transfer sizes defined by the device's negotiated input buffer sizes.[402]

401.CCB 1401
402.CCB 1353

D.6.3.2 Attributes

The client has no attributes.

D.6.3.3 Commands Received

The client receives the cluster-specific response commands detailed in D.6.2.5.

D.6.3.4 Commands Generated

The client generates the cluster-specific commands detailed in D.6.2.4, as required by the application.

D.7 Prepayment Cluster[403]

Note: *The Prepayment Cluster description 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.*

D.7.1 Overview

The Prepayment Cluster provides the facility to pass messages relating to prepayment between devices on the HAN. It allows for the implementation of a system conforming to the set of standards relating to Payment Electricity Meters (IEC 62055) and also for the case where the accounting function is remote from the meter. Prepayment is used in situations where the supply of a service may be interrupted or enabled under the control of the meter or system in relation to a payment tariff. The accounting process may be within the meter or elsewhere in the system. The amount of available credit is decremented as the service is consumed and is incremented through payments made by the consumer. Such a system allows the consumer to better manage their energy consumption and reduces the risk of bad debt owing to the supplier.

In the case where the accounting process resides within the meter, tariff and credit updates are tunneled to the meter from the head end via the ESI. Such messages are out of scope of this cluster. The cluster allows credit status to be made available to other devices on the HAN for example to enable the consumers to view their status on an IHD.[404] It also allows them to select emergency credit if running low and also, where local markets allow, restoring their supply remotely from within the HAN.

403. Incremental Release 1

404. CCB 1570

In the case where the accounting process resides in the head end (Central Wallet scheme), the metering system provides usage information to the head end for it to calculate the state of available credit in the consumer's account. The head end will pass down to the metering system data that will be of use to the consumer, for distribution on the HAN. The head end will also send commands to interrupt or restore the supply depending on the state of the account.

In either case, there will be the need to display credit status and this may be in monetary terms or in energy terms. If running in monetary mode, the units of measure will be defined in the Price Cluster, if in energy terms, the unit of measure will be defined in the Metering Cluster.

The Prepayment Cluster requires three states for the supply status due to safety requirements in certain countries, these are:

- ON
- OFF
- ARMED

The ARMED state is to allow for a remote restoration of the supply that requires action by the consumer (such as pressing a button on the meter or the IHD[405]). This is to ensure the supply is not restored remotely whilst in an unsafe situation.

The three corresponding commands derived from IEC 62055 are:

- *RESTORE*
- *INTERRUPT*
- *ARM*

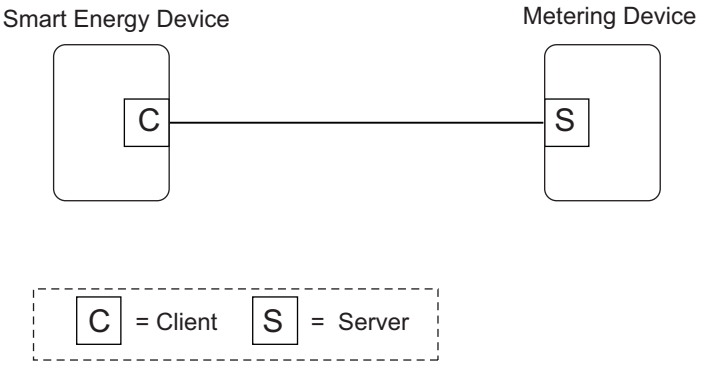

Figure D.50 Prepay Cluster Client Server Example

405.CCB 1570

D.7.2 Server

D.7.2.1 Dependencies

- Support for ZCL Data Types
- 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.
- Use of the Price Cluster is Mandatory when using the Prepayment Cluster in Currency mode.

D.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 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 the following Table D.68.

Table D.68 Payment Attribute Sets

Attribute Set Identifier	Description
0x00	Prepayment Information Set
0x01	Top-up Attribute Set
0x02	Debt Attribute Set
0x03	Supply Control Set

D.7.2.2.1 *Prepayment Information* Attribute Set

The following set of attributes provides access to the standard information relating to a Prepayment meter.

Table D.69 P*repayment Information* Attribute Set

Identifier	Name	Type	Range	Access	Default	Mandatory / Optional
0x00	*Payment Control*	8-bit Bitmap	0x00 to 0xFF	Read only	0x00	M
0x01	*Credit Remaining*	Signed 32-bit Integer	-0x7FFFFFFF to +0x7FFFFFFF	Read only	-	O
0x02	*Emergency Credit Remaining*	Signed 32-bit Integer	-0x7FFFFFFF to +0x7FFFFFFF	Read only	-	O
0x03	*Credit Status*	8-bit Bitmap	0x00 to 0x40	Read only	0x00	O

D.7.2.2.1.1 *Payment Control* Attribute

The *Payment Control* attribute represents the payment mechanisms currently enabled within the Metering Device. Bit encoding of this field is outlined in Table D.70.

Table D.70 *Payment Control* Attribute

Bits	Description
0	Disconnection Enabled
1	Reserved
2	Credit Management Enabled
3	Reserved
4	Credit Display Enabled
5	Reserved
6	Account Base
7	Contactor Fitted

Disconnection Enabled: Indicates whether the metering device is to disconnect the energy supply on expiry of available credit.

Credit Management Enabled: Indicates whether the metering device should manage available credit according to available tariff information.

Credit Display Enabled: Indicates whether the metering device should display the credit status.

Account Base: Indicates whether the metering device is running in Monetary (0) or Unit based (1) units. If Monetary based, the unit of measure is defined in the Price cluster, if Unit based, the unit of measure is defined in the Metering cluster

Contactor Fitted: Indicates whether the metering device is fitted with a Contactor i.e. is capable if disconnecting the energy supply.

D.7.2.2.1.2 *Credit Remaining* Attribute

The *Credit Remaining* attribute represents the amount of credit remaining on the Metering Device. If Monetary-based, this attribute is measured in a base unit of Currency with the decimal point located as indicated by the *Trailing Digits* field, as defined in the Price cluster. If Unit-based, the unit of measure is as defined in the Metering cluster (see sub-clause D.3.2.2.4.1).

D.7.2.2.1.3 *Emergency Credit Remaining* Attribute

The *Emergency Credit Remaining* attribute represents the amount of Emergency Credit still available on the Metering Device. If Monetary-based, this attribute is measured in a base unit of *Currency* with the decimal point located as indicated by the *Trailing Digits* field, as defined in the Price cluster. If Unit-based, the unit of measure is as defined in the Metering cluster (see sub-clause D.3.2.2.4.1).

D.7.2.2.1.4 *Credit Status* Attribute

The *Credit Status* attribute represents the current status of credit within the Metering Device. Bit encoding of this field is outlined in Table D.71.

Table D.71 *Credit Status* Attribute

Bits	Description
0	Credit OK
1	Low Credit
2	Emergency Credit Enabled
3	Emergency Credit Available
4	Emergency Credit Selected
5	Emergency Credit In Use
6	Emergency Credit Exhausted
7	Reserved for Future Use

D.7.2.2.2 Top-up Attribute Set

The following set of attributes provides access to previous credit *top-ups* on a prepayment meter.

Table D.72 Top-up Attribute Set

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x00	*Top up Date/Time #1*	UTCTime		Read only	-	O
0x01	*Top up Amount #1*	Unsigned 48-bit Integer	0x000000000000 – 0xFFFFFFFFFFFF	Read only	-	O
0x02	*Originating Device #1*	8 bits Enumeration	0x00 to 0xFF	Read only	-	O
0x10	*Top up Date/Time #2*	UTCTime		Read only	-	O
0x11	*Top up Amount #2*	Unsigned 48-bit Integer	0x000000000000 – 0xFFFFFFFFFFFF	Read only	-	O
0x12	*Originating Device #2*	8 bits Enumeration	0x00 to 0xFF	Read only	-	O
0x20	*Top up Date/Time #3*	UTCTime		Read only	-	O
0x21	*Top up Amount #3*	Unsigned 48-bit Integer	0x000000000000 – 0xFFFFFFFFFFFF	Read only	-	O
0x22	*Originating Device #3*	8 bits Enumeration	0x00 to 0xFF	Read only	-	O
0x30	*Top up Date/Time #4*	UTCTime		Read only	-	O
0x31	*Top up Amount #4*	Unsigned 48-bit Integer	0x000000000000 – 0xFFFFFFFFFFFF	Read only	-	O
0x32	*Originating Device #4*	8 bits Enumeration	0x00 to 0xFF	Read only	-	O

Table D.72 Top-up Attribute Set (Continued)

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x40	*Top up Date/ Time#5*	UTCTime		Read only	-	O
0x41	*Top up Amount #5*	Unsigned 48-bit Integer	0x000000000000 – 0xFFFFFFFFFFFF	Read only	-	O
0x42	*Originating Device #5*	8 bits Enumeration	0x00 to 0xFF	Read only	-	O

D.7.2.2.2.1 *Top up Date/Time* Attribute

The *Top up Date/Time* attribute represents the time that the credit was topped up on the Metering Device. There are five records containing this attribute, one for each of the last five top-ups.

D.7.2.2.2.2 *Top up Amount* Attribute

The *Top up Amount* attribute represents the amount of credit that was added to the Metering Device during the top up. If Monetary-based, this attribute is measured in a base unit of *Currency* with the decimal point located as indicated by the *Trailing Digits* field, as defined in the Price cluster. If Unit-based, the unit of measure is as defined in the Metering cluster (see sub-clause D.3.2.2.4.1). There are five records containing this attribute, one for each of the last five top-ups.

D.7.2.2.2.3 *Originating Device* Attribute

The *Originating Device* attribute represents the SE device that was the source of the top-up command. The enumerated values of this field are outlined in Table D.77. There are five records containing this attribute, one for each of the last five top-ups.

D.7.2.2.3 Debt Attribute Set

The following set of attributes provides access to information on debt held on a Prepayment meter.

Table D.73 Debt Attribute Set

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x00	*Fuel Debt Remaining*	Unsigned 48-bit Integer	0x000000000000 – 0xFFFFFFFFFFFF	Read only	-	O
0x01	*Fuel Debt Recovery Rate*	Unsigned 32-bit Integer	0x00000000 – 0xFFFFFFFF	Read only	-	O
0x02	*Fuel Debt Recovery Period*	8 bits Enumeration	0x00 to 0xFF	Read only	-	O
0x03	*Non Fuel Debt Remaining*	Unsigned 48-bit Integer	0x000000000000 – 0xFFFFFFFFFFFF	Read only	-	O
0x04	*Non Fuel Debt Recovery Rate*	Unsigned 32-bit Integer	0x00000000 – 0xFFFFFFFF	Read only	-	O
0x05	*Non Fuel Debt Recovery Period*	8 bits Enumeration	0x00 to 0xFF	Read only	-	O

D.7.2.2.3.1 *Fuel Debt Remaining* Attribute

The *Fuel Debt Remaining* attribute represents the amount of Fuel Debt remaining on the Metering Device, measured in base unit of *Currency* with the decimal point located as indicated by the Trailing Digits field, as defined in the Price Cluster.

D.7.2.2.3.2 *Fuel Debt Recovery Rate* Attribute

The *Fuel Debt Recovery Rate* attribute represents the amount of Fuel Debt recovered each *Fuel Debt Recovery Period*, measured in base unit of *Currency* with the decimal point located as indicated by the Trailing Digits field, as defined in the Price Cluster.

D.7.2.2.3.3 *Fuel Debt Recovery Period* Attribute

The *Fuel Debt Recovery Period* attribute represents the period over which each *Fuel Debt Recovery Rate* is recovered. The enumerated values of this field are outlined in Table D.74.

Table D.74 *Fuel Debt Recovery Period* Field Enumerations

Enumerated Value	Recovery Period
0x0	Per Hour
0x1	Per Day
0x2	Per Week
0x3	Per Month
0x4	Per Quarter

D.7.2.2.3.4 *Non Fuel Debt Remaining* Attribute

The *Non Fuel Debt Remaining* attribute represents the amount of Non Fuel Debt remaining on the Metering Device, measured in base unit of *Currency* with the decimal point located as indicated by the Trailing Digits field, as defined in the Price Cluster.

D.7.2.2.3.5 *Non Fuel Debt Recovery Rate* Attribute

The *Non Fuel Debt Recovery Rate* attribute represents the amount of Non Fuel Debt recovered each *Non Fuel Debt Recovery Period*, measured in base unit of *Currency* with the decimal point located as indicated by the Trailing Digits field, as defined in the Price Cluster.

D.7.2.2.3.6 *Non Fuel Debt Recovery Period* Attribute

The *Non Fuel Debt Recovery Period* attribute represents the period over which each *Non Fuel Debt Recovery Rate* is recovered. The enumerated values of this field are outlined in Table D.74.

D.7.2.2.4 Supply Control Set

Note: *There is an intention to move the Supply Control functionality to a separate cluster before it becomes certifiable.*

The following set of attributes provides access to information controlling supply on a Prepayment meter.

Table D.75 Supply Control Attribute Set

Identifier	Name	Type	Range	Access	Default	Mandatory /Optional
0x00	*Proposed Change Provider ID*	Unsigned 32-bit Integer	0x00000000 – 0xFFFFFFFF	Read only	-	O
0x01	*Proposed Change Implementation Time*	UTCTime		Read only	-	O
0x02	*Proposed Change Supply Status*	8 bits Enumeration	0x00 to 0xFF	Read only	-	O
0x03	*Delayed Supply Interrupt – Value Remaining*	Unsigned 16-bit Integer	0x0000 – 0xFFFF	Read/ Write	-	O
0x04	*Delayed Supply Interrupt – Value Type*	8 bits Enumeration	0x00 to 0xFF	Read/ Write	-	O

D.7.2.2.4.1 *Proposed Change Provider ID* Attribute

The *Proposed Change Provider ID* indicates the unique identifier for the commodity provider associated with the proposed change to the supply.

D.7.2.2.4.2 *Proposed Change Implementation Time* Attribute

The *Proposed Change Implementation Time* indicates the time at which a proposed change to the supply is to be implemented. If there is no change of supply pending, this field will be set to zero.

D.7.2.2.4.3 *Proposed Change Supply Status* Attribute

The *Proposed Change Supply Status* indicates the proposed status of the supply once the change to the supply has been implemented. The enumerated values of this field are outlined in Table D.80.

D.7.2.2.4.4 *Delayed Supply Interrupt – Value Remaining* Attribute

In "Central Wallet" type schemes, there is the ability to delay a Remote Disconnect. The *Delayed Supply Interrupt – Value Remaining* attribute represents the value of this delay remaining measured in units defined by the *Delayed Supply Interrupt – Value Type* field.

To initiate a *Delayed Supply Interrupt*, a non-zero value should be written to this attribute.

To cancel an existing *Delayed Supply Interrupt*, 0x0000 should be written to this attribute.

It is expected that the supply will be interrupted when the metering device decrements the value of this attribute from 0x0001 to 0x0000.

Manufacturers may limit write access to certain address or devices. This would prevent, for example, an IHD[406] from setting the attribute to a higher value in order to defeat disconnection.

D.7.2.2.4.5 *Delayed Supply Interrupt – Value Type* Attribute

The *Delayed Supply Interrupt – Value Type* attribute represents the type of units that the *Delayed Supply Interrupt – Value Remaining* field is measured in. The enumerated value for this field shall be taken from the values in Table D.22.

D.7.2.3 Commands Received

Table D.76 lists cluster-specific commands that are received by the server.

Table D.76 Cluster -specific Commands Received by the Server

Command Identifier FieldValue	Description	Mandatory/ Optional
0x00	*Select Available Emergency Credit*	O
0x01	*Change Supply*	O

D.7.2.3.1 *Select Available Emergency Credit* Command

This command is sent to the Metering Device to activate the use of any Emergency Credit available on the Metering Device.

406.CCB 1570

D.7.2.3.1.1 Payload Format

Octets	4	1	1-33[a]	1-17[b]
Data Type	UTCTime	8 bits Enumeration	Octet String	Octet String
Field Name	Command Date/ Time	Originating Device	Site ID	Meter Serial Number

a. CCB 1292

b. CCB 1292

Figure D.51 Format of the *Select Available Emergency Credit* Command Payload

D.7.2.3.1.2 Payload Details

Command Date/Time: A UTCTime field to indicate the date and time at which the selection command was issued.

Originating Device: An 8-bit enumeration field identifying the SE device issuing the selection command, using the lower byte of the Device ID defined in Table 5.14, and summarized in Table D.77.

Table D.77 Originating Device Field Enumerations

Enumerated Value	Device
0x00	Energy Service Interface
0x02	In-Home[a] Display Device

a. CCB 1570

Site ID: An Octet String identifying the location of the metering device (UK MPRN or MPAN). The Site ID is a text string, known in the UK as the M-PAN number for electricity and MPRN for gas and 'Stand Point' in South Africa. These numbers specify the meter point location in a standardised way. The specified field is large enough to accommodate the number of characters typically found in the UK and Europe (16 digits). Although the field is generally numeric, possible alpha-numeric format is catered for by specifying an octet string.

Meter Serial Number: An Octet string providing a unique identification of the metering device

D.7.2.3.2 *Change Supply* Command

Note: *There is an intention to move the Supply Control functionality to a separate cluster before it becomes certifiable.*

This command is sent to the Metering Device to instruct it to change the energy supply.

D.7.2.3.2.1 Payload Format

Octets	4	4	1-33[a]	1-17[b]	4	1	1
Data Type	Unsigned 32-bit Integer	UTCTime	Octet String	Octet String	UTCTime	8 bits enumeration	8-bit BitMap
Field Name	Provider ID	Request Date/ Time	Site ID	Meter Serial Number	Implementation Date/ Time	Proposed Supply Status (after implementation)	Originator ID / Supply Control Bits

a. CCB 1292

b. CCB 1292

Figure D.52 Format of the *Change Supply* Command Payload

D.7.2.3.2.2 Payload Details

Provider ID: An unsigned 32-bit field containing a unique identifier for the commodity provider to whom this command relates

Request Date/Time: A UTCTime field to indicate the date and time at which the supply change was requested.

Site ID: An Octet String identifying the location of the metering device (UK MPRN or MPAN).

Meter Serial Number: An Octet string providing a unique identification of the metering device.

Implementation Date/Time: A UTCTime field to indicate the date at which the supply change is to be applied. An Implementation Date/Time of 0x00000000 shall indicate that the command should be executed immediately. An Implementation Date/Time of 0xFFFFFFFF shall cause an existing but pending *Change Supply* command to be cancelled (the status of the supply will not change but the Proposed Change Implementation Time shall be reset to zero).

Proposed Supply Status (after Implementation): An 8-bit enumeration field indicating the status of the energy supply controlled by the Metering Device following implementation of this command. The enumerated values for this field are outlined in Table D.80.

Originator ID/Supply Control Bits: An 8-bit BitMap where the most significant nibble is an enumerated sub-field identifying the SE device issuing the *Change Supply* command, as defined in Table D.77, and the least significant nibble defines the Supply Control bits, the encoding of which is outlined in Table D.78.

Table D.78 Supply Control Bits

Bits	Description
0	Acknowledge Required
1	Reserved
2	Reserved
3	Reserved

Acknowledge Required: Indicates that a *Supply Status Response* command is to be sent in response to this command.

D.7.2.3.2.3 When Generated

A Head-end or ESI may send an *INTERRUPT*, *ARM* or (if allowed) *RESTORE* command to a metering device.

An IHD[407] may send a *RESTORE* command to a metering device which, if ARMED, should cause the supply to be RESTORED (the IHD[408] may also send an *ARM* command).

The execution of an *INTERRUPT* or *ARM* command may be delayed, as indicated by the Implementation Date/Time field; these commands shall only come from a Head-End via an ESI. A subsequent command with a new Implementation Date/ Time shall override an existing delayed command. A new command with an

407.CCB 1570
408.CCB 1570

Implementation Date/Time of 0x00000000 shall be executed immediately, but shall not cancel an existing delayed command; to override an existing delayed command with a command to be executed immediately, a command to cancel the existing command should first be sent followed by the new command to be executed immediately (see notes on Implementation Date/Time field in D.7.2.3.2.2 for further details).

The addition of credit or selection of Emergency credit shall not cause a delayed *INTERRUPT* command to be cancelled (these will be cancelled by the Head-End and a new supply control command sent down).

A time-delayed *INTERRUPT* command and a Delayed Supply Interrupt may run concurrently; the supply will be interrupted when the first event occurs.

D.7.2.3.2.4 Effect on Receipt

If required, a *Supply Status Response* command shall be returned to the originator when the *Change Supply* command has been successfully executed (see D.7.2.4.1 for further details).

A ZCL response, indicating 'Unauthorized' (NOT_AUTHORIZED), shall be immediately returned to an originator requesting a supply change that is not allowed in the current application.

A ZCL response, indicating 'Unavailable' (UNSUP_CLUSTER_COMMAND), shall be immediately returned to an originator requesting a supply *change* by a metering device that is incapable of carrying out the action (e.g. an INTERRUPT command to a metering device that has no contactor).

A ZCL response, indicating INVALID_VALUE, shall be immediately returned to an originator requesting a supply change containing a non-zero Implementation Date/Time that is less than or equal to the current date/time (i.e. is in the past).

D.7.2.4 Commands Generated

Table D.79 lists commands that are generated by the server.

Table D.79 Cluster -specific Commands Sent by the Server

Command Identifier Field Value	Description	Mandatory/ Optional
0x00	*Supply Status Response*	O

D.7.2.4.1 *Supply Status Response* Command

Note: *There is an intention to move the Supply Control functionality to a separate cluster before it becomes certifiable.*

This command is transmitted by a Metering Device in response to a *Change Supply* command.

D.7.2.4.1.1 Payload Format

Octets	4	4	1
Data Type	Unsigned 32-bit Integer	UTCTime	8-bit Enumeration
Field Name	Provider ID	Implementation Date/Time	Supply Status (after implementation)

Figure D.53 Format of the *Supply Status Response* Command Payload

D.7.2.4.1.2 Payload Details

Provider ID: An unsigned 32-bit field containing a unique identifier for the commodity provider to whom this command relates

Implementation Date/Time: A UTCTime field to indicate the date at which the originating command was to be applied.

Supply Status: An 8-bit enumeration field indicating the status of the energy supply controlled by the Metering Device following implementation of the originating command. The enumerated values for this field are outlined in Table D.80.

Table D.80 Supply Status Field Enumerations

Enumerated Value	Status
0x00	Supply OFF
0x01	Supply OFF/ ARMED
0x02	Supply ON

D.7.2.4.1.3 When Generated

This command is transmitted by a Metering Device to indicate that a *Change Supply* command has been successfully executed. It shall be sent if an acknowledgment is requested in the originating command (see sub-clause D.7.2.3.2).

D.7.3 Client

D.7.3.1 Dependencies

- Support for ZCL Data Types
- 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.

D.7.3.2 Attributes

The client has no attributes.

D.7.3.3 Commands Received

The client receives the cluster-specific response commands detailed in D.7.2.4.

D.7.3.4 Commands Generated

The client generates the cluster-specific commands detailed in D.7.2.3, as required by the application.

D.8 Over-the-Air Bootload Cluster[409]

D.8.1 Overview

The over-the-air bootload cluster provides a common mechanism to manage and serve up upgrade images for devices from different manufacturers in the same network. Servers provide firmware images to clients to download, controlling the timing for downloads and when the actual upgrade to a new version of software is made. Clients periodically query the server for new images and then can download the image at a rate according to their capabilities or policies.

Details for the over-the-air (OTA) bootload cluster are maintained in a separate document, reference [095264r15].

409.CCB 1258

Smart Energy devices may optionally support the over-the-air bootload cluster client or server. If the OTA cluster is implemented by a Smart Energy device then APS encryption on all unicast messages shall be used. Smart Energy devices that implement the client must support ECDSA signature verification of images.

Additionally, over-the-air bootload cluster client devices that are intended to be field upgradeable to Smart Energy 2.0 should support the optional feature “query specific file” in order to potentially receive device specific data necessary for the transition to a Smart Energy 2.0 device.

D.8.2 OTA Bootloading Timing Considerations

The OTA cluster defines the message formatting used to pass device images but does not specify when to use the cluster. The following policies specify how and when to use the OTA cluster such that all devices in an SE network will upgrade at predictable intervals.

1. OTA clients shall perform service discovery to find the OTA server after registration has completed.
2. An OTA client device that does not find an OTA server in the network shall periodically attempt a new discovery once a day.
3. All devices shall query the OTA server at least once a day for information about the next version to upgrade to. Non-sleepy devices in the network may be instructed to begin a new download at any point time via the *Image Notify* command.
4. All client devices may download data as quickly as their capabilities allow, but at a minimum rate of one block per 10 minutes. This means that at a rate of 1 block (50 bytes) per 10 minutes, a 128k file will take 18 days to download

ANNEX

E

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.

E.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.

E.2 Rules and Guideline

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[410] shall prevent mismanaged scheduling of *Overlapping Events* or *Nested Events*. It is recognized Upstream Demand Response/Load Control systems and/or the ESI[411] 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[412] 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).

410.CCB 1072
411.CCB 1072
412.CCB 1072

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*.

E.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 E.1 depicts a representation of those devices in a Smart Energy network.

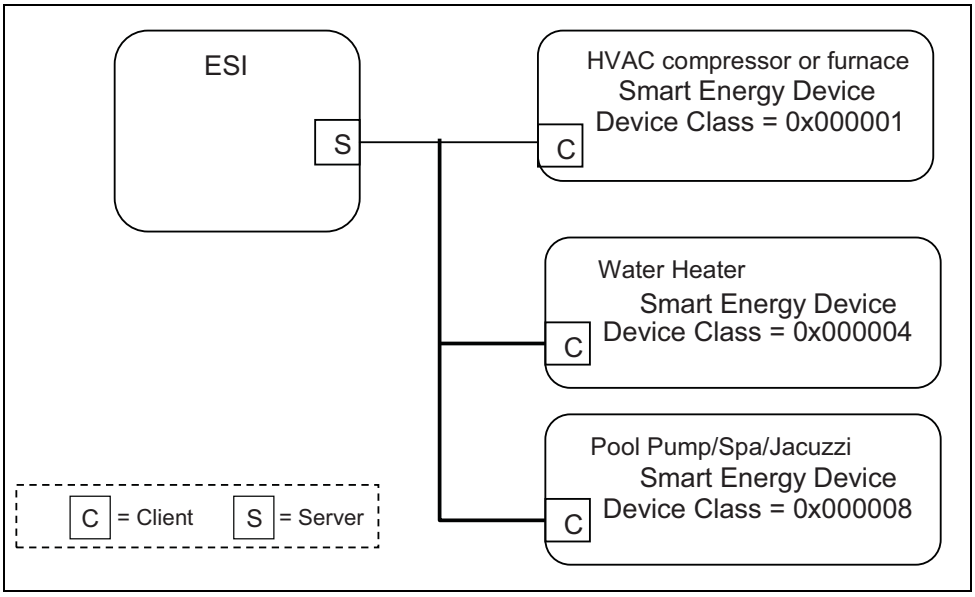

Figure E.1 Smart Energy Device Class Reference Example

E.3.1 Correct Overlapping Events for Different Device Classes

Figure E.2 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.

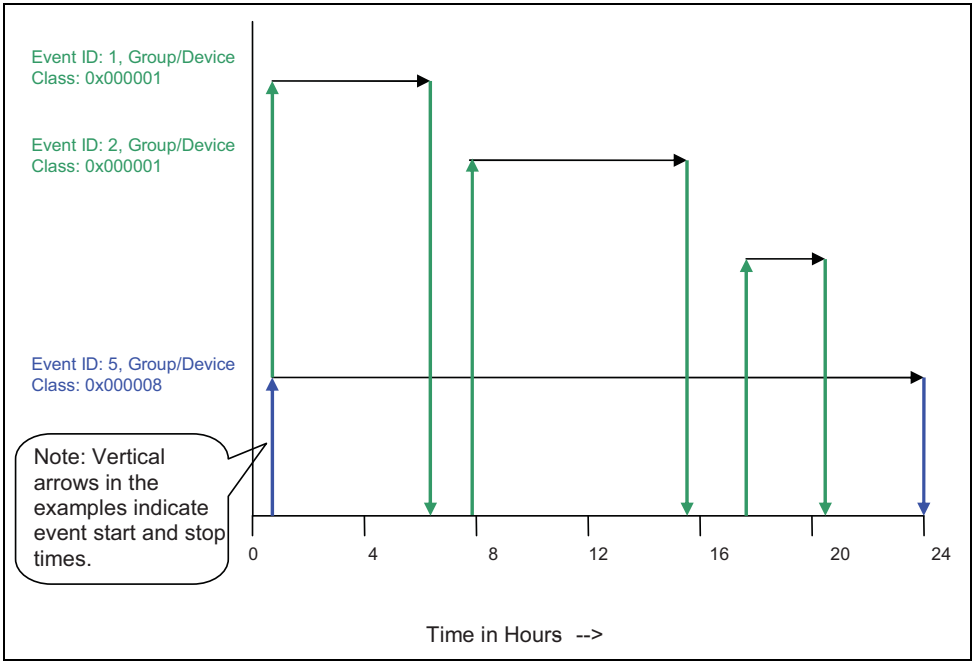

Figure E.2 Correctly Overlapping Events

In Figure E.2, 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.

E.3.2 Correct Superseded Event for a Device Class

Figure E.3 below 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.

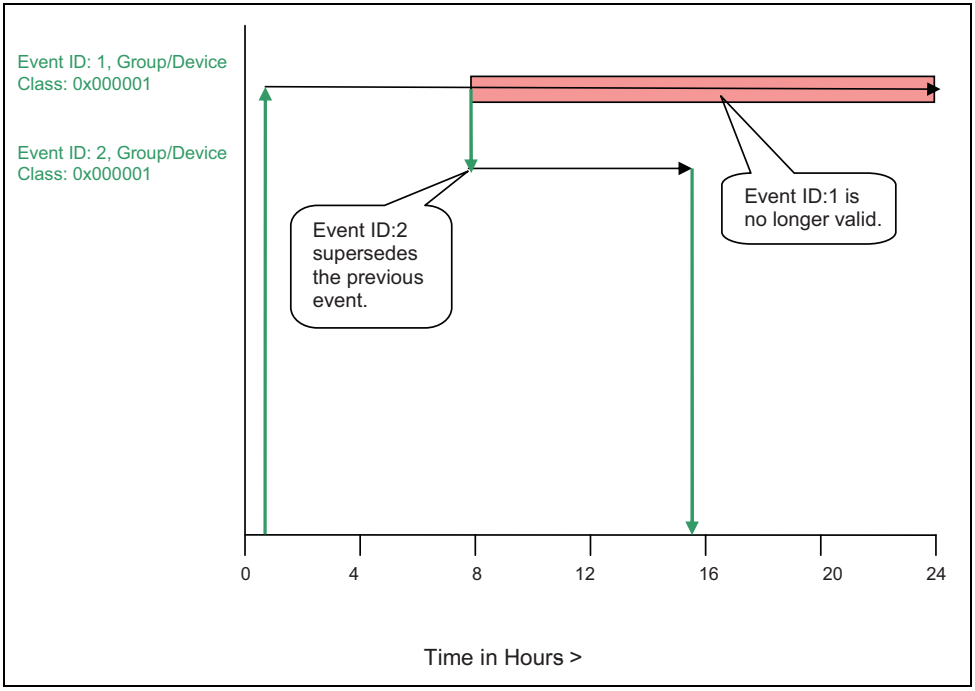

Figure E.3 Correct Superseding of Events

In Figure E.3, 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.

E.3.3 Superseding Events for Subsets of Device Classes

Figure E.4 below 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.

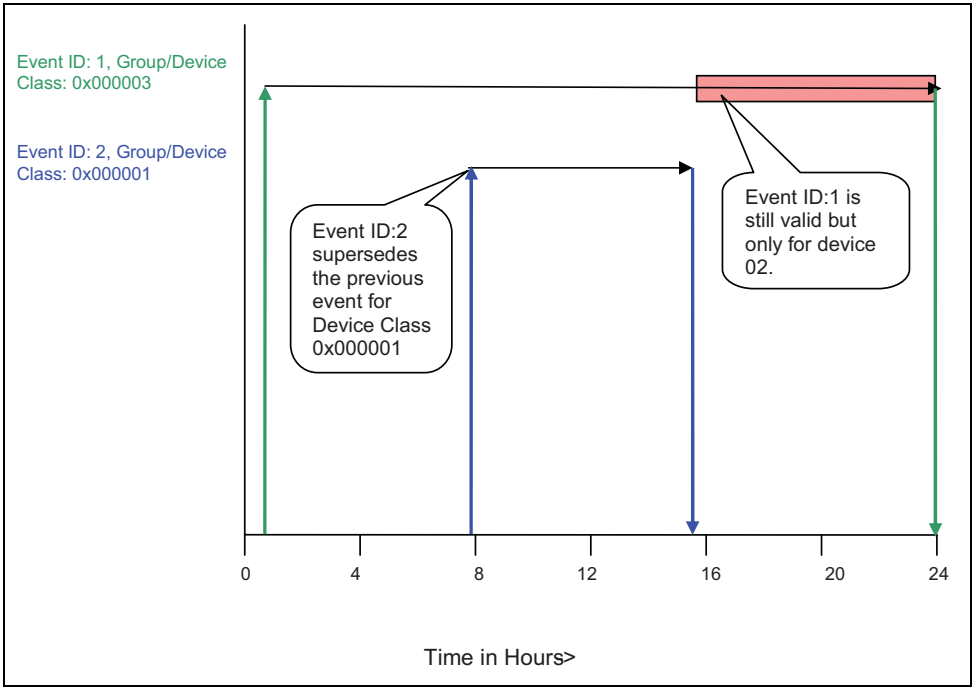

Figure E.4 Superseded Event for a Subset of Device Classes

In Figure E.4, 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.

E.3.4 Ending Randomization Between Events

Figure E.5 below depicts an *Effective End Time* that overlaps a second scheduled DR/LC event for device class of 0x000001 (reference for the BitMap definition).

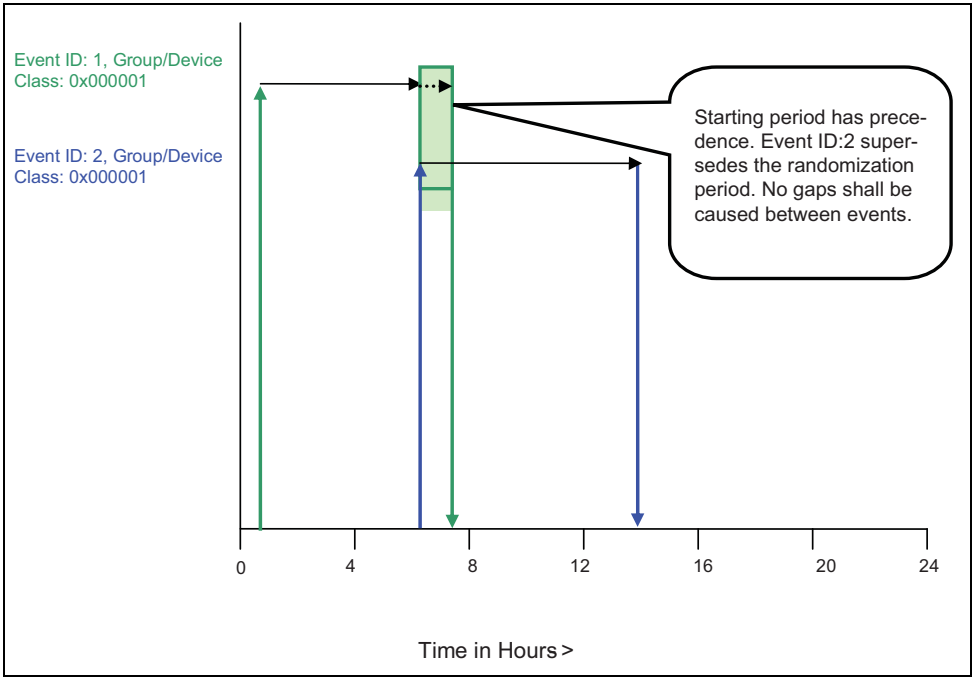

Figure E.5 Ending Randomization Between Events

In Figure E.5, Device Class 0x000001 receives a DR/LC Event ID#1 with an ending randomization setting (please refer to sub-clause D.2.2.3.1.1.1 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.

E.3.5 Start Randomization Between Events

Figure E.6 below depicts an *Effective Start Time* that overlaps a previously scheduled DR/LC event for device class of 0x000001 (reference for the BitMap definition).

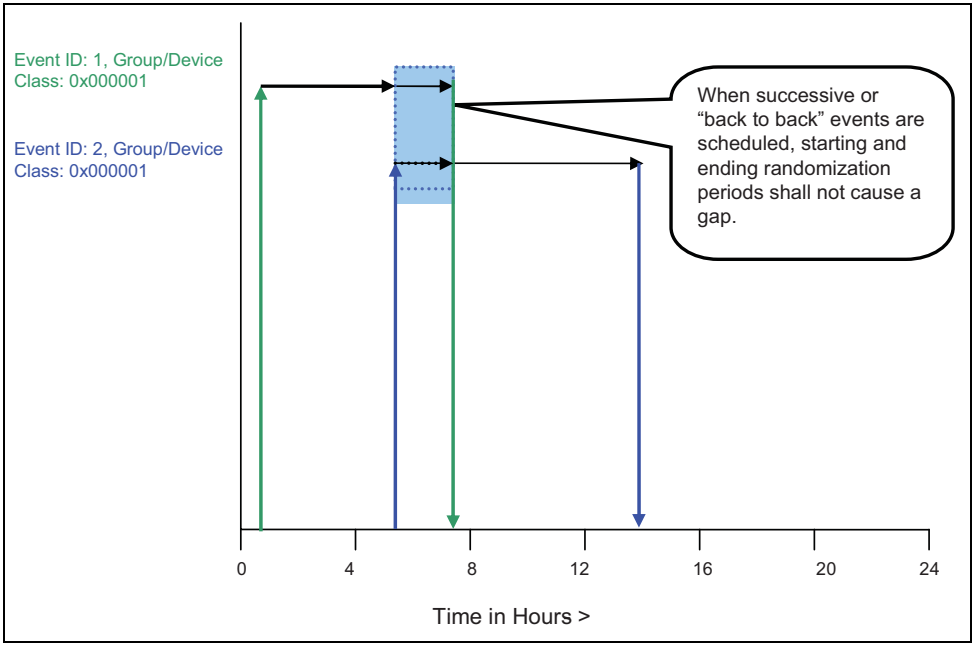

Figure E.6 Start Randomization Between Events

Figure E.6 above, Device Class 0x000001 receives a DR/LC Event ID#1 with an ending randomization setting (please refer to sub-clause D.2.2.3.1.1.1 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.

E.3.6 Acceptable Gaps Caused by Start and Stop Randomization of Events

Figure E.7 below 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.

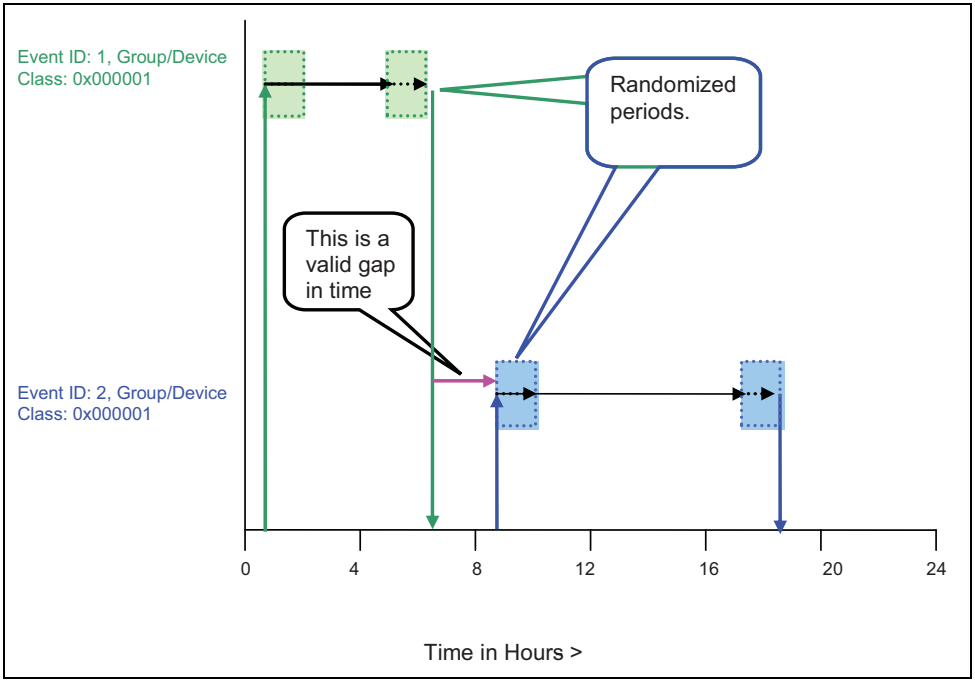

Figure E.7 Acceptable Gaps with Start and Stop Randomization

Figure E.7 above, Device Class 0x000001 receives a DR/LC Event ID#1 with both a starting and ending randomization setting (please refer to sub-clause D.2.2.3.1.1.1 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.

ANNEX

F

JOINING PROCEDURE USING PRE-CONFIGURED TRUST CENTER LINK KEYS

The secure join procedure is detailed as follows:

- The secured joining procedure is as stated in [B3] Section 4.6.3.2.3. The case used in the Smart Energy application is the “Pre-configured trust center link key and address”
- In [B3] Section 4.6.3.2.3.2, in the case of “Pre-configured Trust Center Link Key”, the joining device waits for the APSME-TRANSPORT-KEY. Indication. The frame is encrypted/authenticated with the key-transport key according to the methodologies specified in sections 4.4.1.1 and 4.5.3 of the ZigBee specification r17, which describe the key-transport keys and their association with link keys, in this case the pre-configured trust center link key. The source address will be that of the Trust Center. The key transported will be the NWK Key Key type == 0x01.
- When the trust center sends the tunneled *Transport Key* command, the Extended Nonce bit on the Auxiliary Frame Header must be set to 1 on the Transport Key frame from the Trust Center to the joining child as described in [B3] Section 4.5.1. The Trust Center must also insert its long address into the Source Address field of the Auxiliary Frame Header since that information will be needed at the child to decrypt the *Transport Key* command.
- Sub-clause 5.4 of this document calls out two cases for secured join: pre-configured link keys and temporary link keys. The joining device and trust center perform the same join operation in both cases. The only difference is how the joining device and trust center treat the initial key material (either using it directly as the pre-configured link key or hashing with some data like the long address of the joining device at application level first, see Annex E for this method). From the perspective of the security joining process what happens afterwards is the secure join procedure is the same.

- In either case called out in sub-clause 5.4 of this document, the joining device is authenticated using the [B3] Section 4.6.3.2.3.2 procedure or leaves if the security timeout expires. If authenticated, the key delivered via the APSME-TRANSPORT-KEY.indication in [B3] Section 4.6.3.2.3.2 is the same for either case called out in the AMI specification sub-clause 5.4 (no matter how the application determined the pre-configured link key).

In terms of the message exchange between the child and trust center in performing the secure join procedure, the following is employed:

- Child joining device uses NLME-JOIN.request to parent. Parent sends an APSME-UPDATE-DEVICE.request to the Trust Center on behalf of the child to the Trust Center. APSME-UPDATE-DEVICE.request is transported encrypted/authenticated with the NWK key that the parent has
- Upon receipt at the trust center, the trust center must perform the following processing:
  - Validity check of the child's address to determine if a trust center link key exists between the trust center and the address provided by the joining child.
  - If the child has the trust center as its parent, the APSME-TRANSPORT-KEY.request is sent directly to the child encrypted with the key-transport key derived from the trust center link key known to the child device and the trust center, ELSE
    - —If the child does not have the trust center as its parent, the APSME-TRANSPORT-KEY command frame is encrypted using the key-transport key derived from the trust center link key shared between the child and the trust center.
  - The resulting encrypted payload is sent to the child using the APS *Tunnel* command. The APS *Tunnel* command and its (already encrypted) payload is encrypted using the NWK key from the trust center to the child's parent. On the final hop, the child's parent will perform the following processing according to [B3] Section 4.6.3.7.2:
    - — The parent sends the contents within the APS *Tunnel* command to the child without network layer encryption. The message from the parent to the joining child is an APS encrypted transport key command using the key-transport key derived from the trust center link key.

Here are the details on the message that is routed from the trust center to the joining device's parent via the *Tunnel* command:

- NWK Data Frame (Dest: Parent)
- APS Header (Command)
- APS Command Frame (Tunnel)

- Dest EUI: Child
- Tunnel Payload
- APS Header
- APS Auxiliary Header
- Encrypted Payload
- APS Command Frame (Transport Key)

Here are the details on the message that is routed from the joining device's parent to the joining child:

- NWK Data Frame (added by parent, Dest: child)
- APS Header (from Tunnel Payload)
- APS Auxiliary Header (from Tunnel Payload)
- Encrypted Payload (from Tunnel Payload)
- APS Command Frame (Transport Key)

The message to the child from the parent is identical if the device joins directly to the Trust Center.

As a note on the final hop contents of the payload:

- The last hop of the APME-TRANSPORT-KEY message from parent to joining child has NO network layer encryption, but does have application layer encryption
- Thus: There will be no NWK auxiliary header, but there will be an APS auxiliary header
- The APS auxiliary header will have the Key Identifier Sub-Field set to 0x02 == A key-transport key (see [B3] Section 4.5.1.1.2)
- The APS frame will be encrypted with the key-transport key derived from the pre-configured trust center link key. The pre-configured trust center link key must be part of the apsDeviceKeyPairSet in the AIB of the joining device and also known to the trust center.
- The resulting APS frame from the parent to the joining child is the APS-TRANSPORT-KEY message encrypted with the key-transport key derived from the trust center link key delivered with the key type of key-transport key (0x02).
- Per [B3] Section 4.4.3.2, the KeyType field will be set to (0x01) == Network Key
- The TransportKeyData will be the active network key and sequence number

- The joining device must set the network key and sequence number in its NWK Information Block.
- The device is then joined and authenticated.